

UNIVERSIDADE FEDERAL FLUMINENSE

JOSE RICARDO DA SILVA JUNIOR

Using Massively Parallel Architecture for Media  
Version Control and Fine-Grained Exploratory  
Repository Analysis

NITERÓI

2015

UNIVERSIDADE FEDERAL FLUMINENSE

JOSE RICARDO DA SILVA JUNIOR

**Using Massively Parallel Architecture for Media  
Version Control and Fine-Grained Exploratory  
Repository Analysis**

Thesis presented to the Computing Graduate Program of the Universidade Federal Fluminense in partial fulfillment of the requirements for the degree of Doctor of Science.  
Area: Computer Graphics

Advisor:

ESTEBAN GONZALEZ CLUA

Co-advisor:

LEONARDO GRETA PAULINO MURTA

NITERÓI

2015

# Using Massively Parallel Architecture for Media Version Control and Fine-Grained Exploratory Repository Analysis

Jose Ricardo da Silva Junior

Thesis presented to the Computing Graduate Program of the Universidade Federal Fluminense in partial fulfillment of the requirements for the degree of Doctor of Science.  
Area: Computer Graphics

Approved in December of 2015.

---

Prof. D.Sc. Esteban Gonzalez Clua / UFF (President)

---

Prof. D.Sc. Leonardo Gresta Paulino Murta / UFF

---

Prof. D.Sc. Anselmo Antunes Montenegro / UFF

---

Prof. D.Sc. Débora Christina Muchaluat Saade / UFF

---

Prof. Ph.D. Marcos Roberto da Silva Borges / UFRJ

---

Prof. Ph.D. Marco Aurélio Gerosa / USP

---

Prof. Ph.D. Anita Sarma / OSU

Niterói December 3, 2015.

*“No one save us but ourselves. No one can and no one may. We ourselves must walk the  
path.”*  
*Buddha*



# Acknowledgement

I would like to thank my advisor Esteban Clua for his incentive and support, and mainly his friendship during all my academic life.

My co-advisor Leonardo Murta for all of his lectures about software engineering, advises, friendship, and encouragement during this doctoral course.

To Anita Sarma, for accepting to be my advisor at University of Nebraska, teaching me a lot of interesting subjects during and after the time I spent there.

My family, specially my parents for they unconditional support and love, always providing me with the best they could.

To my wife Giselle, for his patience and comprehension for all the time I was absent, including the long time I lived in US, always giving me strength and love to keep going.

All the people from MediaLab and GEMS group for their support and friendship, specially Daniel Prett for his hard working with Dominoes GUI.

I thank CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) for the financial support during this doctorate.

# Resumo

Sistemas de controle de versão são amplamente utilizados para o controle da evolução de artefatos de software ao longo do tempo, tornando-se uma valiosa fonte de informação para a equipe de desenvolvimento. Visando generalidade e desempenho, esses sistemas se basearam em um modelo de dados simples, que considera arquivos textuais como unidade de versionamento e suas linhas como unidade de comparação. Porém, esse modelo de dados traz dois principais problemas: artefatos binários são tratados de forma opaca e análises mais elaboradas sobre a evolução se tornam custosas. Em relação ao primeiro problema, quando artefatos binários são encontrados, as operações de *diff*, *patch* e *merge* não são aplicadas, fazendo com que esses artefatos sejam armazenados em disco e transportados pela rede por completo em cada revisão realizada. Além disso, também não é possível compreender a diferença entre dois artefatos binários ou combinar artefatos binários editados em paralelo.

Em relação ao segundo problema, dependendo do tamanho do repositório e do seu tempo de vida, a extração e análise das informações não é tarefa fácil. Uma grande quantidade de dados precisa ser reprocessada, compatibilizando o modelo de dados simples usado no versionamento com o paradigma usado pela linguagem de programação do projeto (linhas  $\times$  métodos, por exemplo). Como forma de atenuar este problema, alguns autores fazem a análise em granularidade grossa (no nível de diretórios ou arquivos, por exemplo) ou restringem o tamanho do histórico a ser analisado. Todavia essas restrições tornam o resultado menos preciso.

Dado que ambos os problemas ocorrem em especial pela demanda computacional do processamento de artefatos binários e das tarefas de análise de histórico, esta tese lança mão do poder computacional de GPU. Para tal, é proposta uma abordagem utilizando GPU que possibilite o versionamento sobre artefatos binários (imagem e vídeo especificamente) através de operações especializadas de *diff*, *patch* e *merge*. Além de possibilitar uma acurácia maior durante a exploração de repositórios compostos por este tipo de artefatos e permitir o desenvolvimento em paralelo, a nossa abordagem também reduz o espaço em disco necessário para o armazenamento desses artefatos, reduzindo também, como consequência, a largura de banda necessária para a transferência de repositórios em sistema de controle de versão distribuído. Por outro lado, nós também apresentamos uma abordagem em GPU com o objetivo de possibilitar a exploração de conhecimento sobre repositórios que contêm grande quantidade de dados, permitindo análises em granularidade fina, como métodos e classes.

**Palavras Chave:** Gerenciamento de Software, Sistema de Controle de Versão, Artefato de Imagem, Artefato de Vídeo, Especialização, Dependências, CUDA, GPU.

# Abstract

Version control systems are widely used for controlling software artifacts evolution through time, becoming a valuable source of information to the development team. Aiming at generality and performance, these systems were conceived to work with a simple data model that considers textual artifacts as unit of versioning and their lines as unit of comparison. However, this data model presents two main drawbacks: binary artifacts are treated as opaque data and a more elaborated analysis over software evolution becomes expensive.

In relation to the first problem, when binary artifacts are found, the *diff*, *patch*, and *merge* operations are not performed over them, leading to expensive requirements for storing and transferring them. Besides that, it is not possible to understand the differences or merge two binary artifacts modified in parallel.

Regarding the second problem, depending on the repository size and its lifetime, extracting and analyzing information are not trivial tasks. A huge amount of data needs to be reprocessed, matching the simple model used during versioning with the programming language paradigm used by the project (e.g., lines  $\times$  methods). In order to attenuate this problem, some authors perform a coarse grain analysis (e.g., directory or file), or restrict the portion of the repository history that can be analyzed. Unfortunately, such restrictions produce less accurate results.

Given that the presented problems occur specially due to the computational processing requirements for dealing with binary artifacts and analyzing repository history, this thesis makes use of the GPU's computational power to tackle both problems. We propose a GPU approach that allows binary artifacts versioning (specially image and video) through specialized *diff*, *patch*, and *merge* operations. Besides enabling a more accurate exploration of repositories composed by such kind of artifacts and making possible parallel development over them, our approach also reduces their storage and network bandwidth requirements. On the other hand, we also present a GPU approach for enabling knowledge exploration over repositories that contain huge amounts of data, allowing a fine grain analysis considering methods and classes.

**Keywords:** Software Management, Version Control System, Image Artifact, Video Artifact, Expertise, Dependencies, CUDA, GPU.

# Contents

<b>List of Figures</b>	<b>xi</b>
<b>List of Tables</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	3
1.2 Goal . . . . .	5
1.3 Organization . . . . .	7
<b>2 Version Control Systems</b>	<b>9</b>
2.1 Introduction . . . . .	9
2.2 Basic concepts . . . . .	10
2.3 VCS topology . . . . .	13
2.3.1 Centralized topology . . . . .	14
2.3.2 Distributed topology . . . . .	14
2.4 Base VCS operations . . . . .	15
2.4.1 <i>Diff</i> . . . . .	16
2.4.2 <i>Patch</i> . . . . .	19
2.4.3 <i>Merge</i> . . . . .	19
2.5 Final considerations . . . . .	20
<b>3 Image-aware version control</b>	<b>22</b>
3.1 Introduction . . . . .	22

3.2	Motivational example: JECRIPE project . . . . .	23
3.3	Digital image . . . . .	24
3.4	Proposed approach . . . . .	25
3.4.1	The <i>diff</i> operation . . . . .	26
3.4.2	The <i>Patch</i> operation . . . . .	27
3.4.3	The <i>Merge</i> operation . . . . .	29
3.4.4	Image Processing Techniques . . . . .	31
3.4.5	Data structure . . . . .	34
3.5	Evaluation . . . . .	36
3.5.1	Repository outgrowth . . . . .	36
3.5.2	Performance measurement . . . . .	41
3.6	Threats to validity . . . . .	43
3.7	Related work . . . . .	43
3.8	Final considerations . . . . .	46
<b>4</b>	<b>Video-aware version control</b>	<b>47</b>
4.1	Introduction . . . . .	47
4.2	Digital video . . . . .	48
4.3	<i>VIMUFF</i> <i>diff</i> , <i>patch</i> , and <i>merge</i> operations . . . . .	50
4.3.1	<i>Diff</i> operation . . . . .	50
4.3.1.1	Data structure . . . . .	60
4.3.2	<i>Patch</i> operation . . . . .	61
4.3.3	<i>Merge</i> operation . . . . .	62
4.4	Evaluation . . . . .	65
4.4.1	Storage space . . . . .	65
4.4.2	Processing time . . . . .	67
4.5	Threats to validity . . . . .	70

---

4.6	Related work . . . . .	70
4.7	Final considerations . . . . .	71
<b>5</b>	<b>Exploratory data analysis of software repositories</b>	<b>72</b>
5.1	Introduction . . . . .	72
5.2	Dominoes . . . . .	74
5.2.1	Architecture . . . . .	75
5.2.2	Dominoes tiles . . . . .	77
5.2.3	Specialized operations . . . . .	79
5.2.4	Dominoes GUI . . . . .	81
5.2.4.1	Design Rationale . . . . .	81
5.2.4.2	Interface . . . . .	83
5.3	Examples of Dominoes applicability . . . . .	86
5.3.1	Calculating dependencies using Dominoes . . . . .	86
5.3.2	Expertise identification using Dominoes . . . . .	88
5.3.2.1	Granularity Matters . . . . .	88
5.3.2.2	Time matters . . . . .	93
5.4	Case studies . . . . .	93
5.4.1	Dependency identification . . . . .	94
5.4.2	Expertise identification . . . . .	96
5.4.3	Expertise evolution . . . . .	100
5.4.4	Performance . . . . .	102
5.5	Usability evaluation . . . . .	104
5.5.1	Methodology . . . . .	105
5.5.1.1	Scenarios . . . . .	105
5.5.1.2	Participants . . . . .	106
5.5.1.3	Study Design . . . . .	106

5.5.1.4	Coding . . . . .	108
5.5.2	Results . . . . .	108
5.5.2.1	How is the influence of the use of derived tiles for getting the right answer? . . . . .	109
5.5.2.2	Is it important to check the data produced along the ex- ploration? . . . . .	112
5.5.2.3	What is the influence of relationships representation dur- ing exploration? . . . . .	114
5.5.2.4	What is the behaviour of participants after changing their exploratory path due to mistakes? . . . . .	115
5.5.2.5	What are the barriers to start using Dominoes? . . . . .	116
5.5.3	Discussion . . . . .	117
5.6	Threats to validity . . . . .	118
5.7	Related work . . . . .	119
5.8	Final considerations . . . . .	123
<b>6</b>	<b>Conclusion</b>	<b>125</b>
6.1	Contributions . . . . .	125
6.1.1	<i>Diff</i> , <i>patch</i> , and <i>merge</i> over image and video . . . . .	126
6.1.2	Repository analysis . . . . .	126
6.2	Limitations . . . . .	127
6.3	Future work . . . . .	129
	<b>References</b>	<b>132</b>
	<b>Appendix A – Systematic mapping on version control over multimedia artifacts</b>	<b>140</b>
A.1	Introduction . . . . .	140
A.2	Systematic mapping protocol . . . . .	141
A.2.1	Search methodology and protocol evaluation . . . . .	142

---

A.2.2	Selection criteria . . . . .	144
A.2.3	Extraction and data storage procedure . . . . .	145
A.3	Search procedure . . . . .	145
A.4	Results analysis . . . . .	147
A.5	Complete list of papers returned by the search expression . . . . .	150
A.6	Data collected from the selected papers . . . . .	187
 <b>Appendix B - Graphics Processing Unit</b>		<b>194</b>
 <b>Appendix C - Dominoes Performance Comparision</b>		<b>197</b>



# List of Figures

2.1	Elements versioned by a VCS [86]. . . . .	11
2.2	A branch for implementing a new function in the project. In (a) a new branch is created (named <i>func-Z</i> ) derived from the main line (named <i>master</i> ). In (b) modifications were made in both <i>master</i> and <i>func-Z</i> branches. Finally, in (c) the branch is merged to the main line of development. . . . .	12
2.3	VCS history composition (adapted from [86]). . . . .	13
2.4	CVCS (a) vs DVCS (b) topology [85]. . . . .	13
2.5	<i>Delta</i> storage strategies: Complete, forward, and reverse [85]. . . . .	17
2.6	The <i>diff3</i> operation used to identify changes on parallel development. . . . .	18
2.7	The <i>merge</i> operation aims at conciliating modifications from variants <i>B</i> and <i>C</i> , generating version <i>D</i> . . . . .	20
3.1	Study of different artifact types. . . . .	23
3.2	Mona Lisa using an alpha blending effect. . . . .	24
3.3	Applying a <i>diff</i> operation to images to obtain their <i>delta</i> . In the leftmost image a scene without Wally. In the center one, the same scene with Wally, and on the right image, the <i>delta</i> between them. . . . .	27
3.4	The left image shows a scene without Wally. In the center one, the <i>delta</i> from one revision of a scene where Wally is shown, and the one on the right has the reconstruction of the scene. . . . .	28
3.5	Steps needed to execute <i>merge</i> . . . . .	30
3.6	Applying a <i>merge</i> operation. . . . .	31
3.7	The <i>delta</i> generated after applying a global transformation to an image. . . . .	32
3.8	Homogeneous bi-dimensional rotational matrix applied to an image. . . . .	33
3.9	Mapping characteristics of images for extracting a rigid transformation. . . . .	33

3.10	Data structure describing information stored for <i>Image A</i> . A table is used to locate revision data in different packages. Black hashed blocks represent <i>delta</i> size for each revision while white blocks store its matrix. . . . .	36
3.11	<i>IMUFF</i> toolkit used to perform <i>diff</i> , <i>patch</i> and <i>merge</i> operations over image artifacts. Available at: <a href="https://github.com/gems-uff/gemuff">https://github.com/gems-uff/gemuff</a> . . . . .	37
3.12	Image distribution in terms of size. . . . .	38
3.13	Image area change transformation ( <i>IMUFF V1</i> and <i>V2</i> overlap). . . . .	38
3.14	Applying global transformations: filter and rotation (Git, HG, and SVN overlap). . . . .	39
3.15	Applying horizontal flipping and evenly distributed transformations (Git, HG, and SVN are overlapped). . . . .	40
3.16	Comparison of <i>diff</i> , <i>patch</i> and <i>merge</i> operations as run by the CPU and GPU for four image sizes. Vertical axis is shown in $\log_{10}$ . . . . .	42
3.17	Time needed to make a check-in operation in the evenly distributed transformations case. . . . .	43
4.1	Common resolutions for video artifacts (width $\times$ height) [68]. . . . .	49
4.2	Activity diagram for processing a <i>diff</i> in <i>VIMUFF</i> . . . . .	50
4.3	Identification and list sequence generation in <i>VIMUFF</i> . . . . .	51
4.4	The difference between two video's sequence. Nodes colored in yellow are common to both videos, while red nodes represents frames not in <i>video 2</i> (removed) and green nodes represents frames just in <i>video 2</i> (added). . . .	52
4.5	<i>Diff</i> detection by <i>VIMUFF</i> . In (a) frames were added while in (b) frames were removed, and finally in (c) frames were changed. . . . .	54
4.6	DCT processing over two different images. The image in (a) has a small slope between tones changes while in (b) tones change abruptly. Images taken from [66]. . . . .	55
4.7	The difference between two video's sequence. Nodes colored in yellow are common to both videos, while red nodes represent frames not in <i>video 2</i> (removed) and green nodes represent frames just in <i>video 2</i> (added). Cyan nodes represent frames that suffered modifications. . . . .	58

4.8	Expansion activity diagram for both <i>DCT Hash</i> and <i>Similarity Diff</i> processes. Yellow activities are done in GPU. . . . .	63
4.9	Example of how <i>delta</i> is organized. In (a) video 1 suffered a modification, where yellow, cyan, and green represent LCS, modified, and added frames, respectively. In (b) the data structure that represents these operations. . .	64
4.10	Merge of two videos <i>B</i> and <i>D</i> descending from <i>A</i> and producing final video <i>E</i> . . . . .	64
4.11	Comparing using <i>delta</i> and not using it for storing video artifacts. . . . .	66
4.12	Time spent for each step during a <i>diff</i> processing using GPU. . . . .	67
4.13	Execution time for processing the <i>DCT Hash</i> and <i>Similarity Diff</i> between two versions of a video using GPU and CPU. . . . .	68
4.14	Execution time running a <i>patch</i> operation. . . . .	68
4.15	Time spent for each step during a <i>merge</i> processing using GPU. . . . .	69
4.16	Execution time for processing the <i>DCT Hash</i> and <i>Similarity Diff3</i> during a <i>merge</i> operation using GPU and CPU. . . . .	69
5.1	Dominoes architecture. . . . .	75
5.2	A set of panes depicting Mamta's interactions with Dominoes. The video of the usage scenario is at: <a href="https://github.com/gems-uff/dominoes">https://github.com/gems-uff/dominoes</a> . . . . .	84
5.3	Support, Confidence, and Lift calculated from previous scenario. . . . .	87
5.4	[developer file time] tile with layers in the back denoting recency. . . . .	93
5.5	Relation among confidence for various support threshold. The leftmost chart considers a threshold of 10, while the middle uses 20, and finally the rightmost uses 30. . . . .	95
5.6	Developer breadth expertise for file <i>EmbedConnection.java</i> . . . . .	101
5.7	Developer breadth expertise for the whole project. . . . .	102
5.8	Participants' characteristics. . . . .	107
5.9	Experiment workflow performed for each participant. . . . .	107
5.10	Participants' action map for the experiment. . . . .	108
5.11	Time (in minutes) taken by each participant during the experiment. . . . .	110

5.12	Total of derived tiles and unique derived tiles, and right answer. . . . .	110
5.13	Actions performed by P2 on scenario 2. . . . .	113
5.14	Relationship about deviation, moving forward, and backtracking. The numbers over the lines represents tiles / minute. . . . .	115
5.15	Dominoes feedback from the participants. . . . .	116
5.16	Word cloud chose by participants from the Microsoft Reaction Card. . . .	117
A.1	Papers returned by the digital library. . . . .	145
A.2	Papers selected through the first filter. . . . .	146
A.3	Papers selected through the second filter. . . . .	147
A.4	Paper classification by multimedia artifacts. . . . .	147
A.5	Techniques used for multimedia artifact management. . . . .	148
A.6	Tool support offered. . . . .	148
A.7	VCS integration availability. . . . .	149
A.8	User intervention degree necessary for applying <i>diff</i> , <i>patch</i> , and <i>merge</i> operations over artifacts. . . . .	149
A.9	Dependent variables used to evaluate the paper. . . . .	150
B.1	GPU and CPU processing capacity [28]. . . . .	195
B.2	A common GPU architecture [94]. . . . .	195
C.1	Transposition operation using different matrix sizes in both CPU and GPU.	198
C.2	Support operation using different matrix sizes in both CPU and GPU. Vertical axis are presented in $\log_{10}$ scale. . . . .	198
C.3	Confidence operation using different matrix sizes in both CPU and GPU. .	199
C.4	Processing transpose, support, and confidence operations on GPU using different matrices sizes. . . . .	199

# List of Tables

3.1	Matrix of a $3 \times 3$ pixels image with a white dot in its center. . . . .	25
3.2	Evolution of a repository size for a specific file over three commits. . . . .	40
3.3	Speedup in running <i>diff</i> , <i>patch</i> and <i>merge</i> operations using CPU and GPU (time in milliseconds). . . . .	41
4.1	Examples of calculating the Hamming distance. In the first row, a binary string is used while in the second a decimal string is used. Finally, the third string shows an alphabet [103]. . . . .	57
4.2	Properties of the videos used during the experiments. . . . .	65
4.3	Frames removed, added and changed as well as the <i>delta</i> size for each case.	66
5.1	Commits made by developers. . . . .	86
5.2	Methods changed for commit. . . . .	87
5.3	Developer $\times$ File. . . . .	89
5.4	Standard Score. . . . .	90
5.5	Developer $\times$ Method (DM matrix). . . . .	91
5.6	Developer $\times$ Method z-score. . . . .	91
5.7	Expertise and z-score at file level. . . . .	92
5.8	Top 5 logical dependencies in terms of high support and biggest confidence difference. . . . .	94
5.9	Absolute and z-score ED for CreateAliasConstantAction.java. . . . .	96
5.10	Absolute and z-score EBD for CreateAliasConstantAction.java. . . . .	97
5.11	Top expert developers at file <i>EmbedConnection.java</i> by ED and EBD <sup>M</sup> .	98
5.12	Top 10 expert developers by ED, EBD <sup>F</sup> , and EBD <sup>M</sup> . . . . .	99
5.13	Processing time (in seconds) for calculating 3D tile for EBD in the project.	104

---

5.14	Coding used for participants' actions. . . . .	109
5.15	Checkpoints performed by the participants in different situations. . . . .	114
A.1	Complete list returned by systematic mapping review. . . . .	150

# Chapter 1

## Introduction

One of the key responsibilities of Configuration Management is to perform version control of software artifacts during software development and maintenance. Version control is the process of organizing, coordinating, and managing the development of evolving artifacts, which can be defined as a series of incremental refinements, and is considered to be an important task of digital content management [40, 33]. As an example, we can cite the evolution of a software product, which can become more complex over time as bugs are fixed and new functionalities are added. In this scenario, evolving artifacts are updated to produce the next versions of the software product in the evolutionary process. In many development areas, the preservation of intermediate revisions of these evolving artifacts is very important to allow rollback operations in the case of an undesired change or simply to check modifications between versions of the software.

Version control systems (VCS) are automated tools that assist the management of these evolving artifacts, providing functionalities that allow users to track intermediate revisions of artifacts and allowing distributed and parallel development. A key element in VCS is the repository, which contains versioned data stored on disk, varying from a single file to a complete source tree. This data exists on two levels, which are called the raw data and the internal model. The former consists of file and directories on the hard drive while the latter represents how these files and directories are maintained by a VCS [100].

In order to process artifacts, VCS normally rely on three basic operations: *diff*, *patch*, and *merge*. *Diff* is used to extract the differences, or *deltas*, between two versions of an artifact. On the other hand, the *patch* is used to generate a specific version of an artifact based on an existing version and a *delta*. The history maintained by a VCS consists of a base version and the sequence of *deltas* that led to the current (or initial) state of

the internal model. Applying a *patch* using a specific *delta* on a repository consists of extending the history by updating the internal model of the repository as well as the artifact itself [100]. Keeping a history with versioned artifacts by a VCS allows a better comprehension of the project.

Both *diff* and *patch* allow a more efficient data storage as just the difference between two versions of an artifact is saved. At the same time, network bandwidth requirement is reduced, since only the *delta* is moved between repositories. Additionally, the *delta* provides a better comprehension of modifications performed over artifacts. Finally, the *merge* operation is used to reconcile two parallel modifications of an artifact, both based on a common ancestor. In this case, each modification is performed on different branches of a repository and merged lately. Branches are isolated areas tracked by a VCS where artifacts can be modified. Each branch is kept apart from the others. While performing a merge, a conflict may occur, requiring manual intervention. Such conflicts can occur, for instance, when the same line of a source code is modified in parallel.

Over 40 years of VCS existence, processing time and storage space heavily influenced VCS design. Conventional VCS divide artifacts into two groups: textual and binary. Textual artifacts are processed by using their lines as unit of comparison [84], allowing *diff*, *patch*, and *merge* operations to be performed on them. The line-based approach is a very useful technique due to its efficiency and scalability [81], in addition to generality. This leads to a better comprehension of modifications (when compared to looking at the original artifacts), reduction of storage space and network bandwidth due to the use of *delta*, and parallel work due to the use of *merge*. On the other hand, binary artifacts do not have any kind of processing by VCS (or have only an inefficient block-based *diff* and *patch* support performed by a binary *diff*), being considered as an opaque data and modeled as a blob of bits. Allowing *diff*, *patch*, and *merge* operations for binary artifacts would lead to processing a great amount of data. Due to the fact that binary artifacts do not produce *deltas*, the comprehension of modifications over them becomes difficult, the required storage space and network bandwidth increase, as each version is saved as a whole (or with an inefficient block-based *delta*), and parallel work over them is not possible. Finally, regarding storage space, the delta storage employed by a VCS can be backward and forward. The *forward delta* algorithm stores the first version of an artifact, being necessary to apply a *delta* on each version to reach more recent versions. On the other hand, the *backward delta* algorithm stores the most recent version of an artifact, requiring the application of a *delta* to recover previous versions.



## 1.1 Motivation

Due to limitations presented on VCS to process a large amount of fine-grained data, two negative consequences are observed: (1) the lack of *diff*, *patch*, and *merge* support for binary artifacts, and (2) inability to perform on-line exploratory repository analysis over the project history.

Many projects are largely composed by binary artifacts (such as image and video) both in terms of number of artifacts or the space occupied by them. Game projects, for instance, can have a large amount of image files to serve as textures. The same can be told for projects in the movie industry, where the number of multimedia artifacts tends to be high. The lack of a tailored support for binary data can cause several issues. For example, existing general purpose VCS, such as Git, adopt a state-based model to store different revisions of these files without any delta information to its predecessor, thus requiring more storage space and making hard to deduce changes between revisions [59] in a high level manner.

Allowing binary artifacts to be processed by VCS requires their internal structures to be considered. This leads to the creation of specialized VCS for these artifacts or the customization of existing VCS to deal with these artifacts more efficiently. These approaches usually require specific *diff*, *patch*, and *merge* operations to manipulate each type of binary artifact. As an example, Odyssey-VCS [84] is a VCS that works specifically with UML models, allowing operations to be consistently executed with this type of artifact.

When considering image and video artifacts, a large amount of data needs to be processed for allowing specialized operations of *diff*, *patch*, and *merge*. In fact, using a VCS for the image context requires processing all pixels for a given image in order to control its evolution. Processing these pixels may become a very time consuming operation, depending on the image size. As an example, processing a common **1,024** width by **1,024** height image requires the individual analysis of **1,048,576** pixels ( $1,024 \times 1,024$ ), which can use a considerable computing time, specially when such image has tens or hundreds of revisions. The amount of time necessary to obtain specific versions of an image may impact the team productivity, which normally does not happen when dealing with line-based artifact. For a video artifact, for instance, the number of images, called frames, can be in the order of hundreds of thousands.

Some related work [49, 102, 26] record user actions while using an image editing soft-

ware. The visualization of such user action histories can aid distinguishing modifications between two versions of an image and is a popular topic among researches [70, 51, 67, 55]. The main problem is that this approach relies on knowing which tool has been used to edit the image artifact, which makes impossible its integration with a VCS. This approach is traditionally implemented inside image and video editing tools. Another approach [58] tries to detect objects in images using computer vision techniques and track their modifications between two versions of an image, but not aimed at VCS. However, algorithms for computer vision are complex, requiring a considerable amount of processing time. Besides that, it cannot be used solely for representing difference, as computer vision algorithms are not always precise. In relation to video artifacts, a few tools exist (Adobe Premiere<sup>1</sup>, Sony Vegas<sup>2</sup>), most of them related to manually merging two videos without any relation with VCS. In this case, the user is responsible for producing the merged video by selecting how the frames will be combined. To be used as a VCS, the video should be automatically merged, requiring user intervention just in case of conflicts.

When working on software projects, developers often need to answer numerous questions, such as: *“which other methods do I need to edit if I make this change?”*; *“who was the person that last edited this method?”*; *“who do I need to coordinate my changes with?”*; *“who is the expert in a specific file?”* and so on [44]. Since software development leaves behind activity logs (e.g., commits recorded in the version control system), it is possible to answer some of these questions by doing a repository analysis.

However, the analysis of such software repository data is not trivial, especially when there is an extensive amount of data that is accrued over the project’s lifecycle [93]. A large software project may comprise thousands of files, with hundreds of developers leading to thousands of commits per month, making it difficult to process this data at interactive rates.

Some related work deal with repository analysis by using some scoping strategy based on: (1) filtering the data [82, 46], (2) performing coarse-grained analysis [23, 44, 93], and (3) overlooking evolution [79, 4, 65, 91, 96]. In the first case, the approaches either scope the amount of data that is processed or the time period over which processing is performed, usually leading to imprecise results. In the second case, tools often analyze data at a coarse-grain level, such as files, without considering which parts of this file has been modified for calculating expertise. Finally, some approaches consider the entire history of the project at once to recommend experts, overlooking the fact that artifacts

---

<sup>1</sup>Website: <http://www.adobe.com/br/products/premiere.html>

<sup>2</sup>Website: <http://www.sonyvegas.com.br>

evolve over time and that developers may change their roles. Finally, all approaches work offline when not scoping the data.

## 1.2 Goal

As observed, some tasks in software engineering cannot be performed due to the amount of data that needs to be processed. Specifically for VCS, giving support for processing image and video artifacts demands analyzing a great amount of data to allow *diff*, *patch*, and *merge* operation to be applied over them in a similar way that is done for text-based artifact. On the other hand, extracting precise information from VCS requires processing fine grain data, such as methods and classes, from a repository that can be composed of hundreds of thousands of versions during a long-living project.

The goal of this thesis consists on allowing such tasks to be performed, opening a new realm of possibilities as discussed before. For this, we conceived, implemented, and evaluated algorithms solving the aforementioned problems, offering VCS support for: (1) *diff*, *patch*, and *merge* operations over image and video artifacts and (2) fast processing of large repositories for knowledge extraction.

Providing a VCS the capability to manage multimedia artifacts, specifically for image and video, requires specialized operations for performing *diff*, *patch*, and *merge*. This kind of support allows a more precise understanding of how image and video artifacts evolve, showing the exact difference between versions. In addition to help developers understanding the changes, we also employ global transform detection over images, such as a rotation. Moreover, our approach for video artifacts also aims at detecting modifications performed on the video, such as subtitle addition, instead of just detecting addition and removal of frames, as performed by the vast majority of VCS. With this approach we provide an additional reduction of the storage space in the repository, since only the exact difference between two versions of the artifact need to be stored. As a natural consequence, the bandwidth requirement when using a distributed VCS is supposed to be lower as the repository becomes smaller.

In order to make our approach viable to be performed on a reasonable amount of time, we employed a high parallel architecture of GPU. Differently from other architectures, which normally rely on infrastructure of multiples nodes connected in a network and demand some level of authentication, the potential of GPU can be used in a single personal computer. By employing such high parallel architecture for repository exploration, we

aim at allowing analyses at any granularity (e.g., methods, classes, files), as well as over unrestricted length of history. Additionally, in order to ease the user exploration in software development process, we adopted a metaphor of dominoes pieces to represent relationships. By combining these pieces, new relationships are produced, which can be further combined. By relaxing constraints related to the amount of history, we believe that more accurate information can be constructed during analysis.

The main contribution of this thesis is then related to making possible some software engineering tasks that are normally ignored due to the their cost of processing. To do so, we use a massively parallel architecture of GPU aimed at reducing the time required for processing these tasks. In this thesis, we solved software engineering problem specifically for VCS. Algorithms and techniques especially designed to work on this highly parallel architecture were conceived, implemented, and evaluated by this thesis.

In order to evaluate our approach, some experiments were performed. In relation to image artifacts, we evaluated:

- The repository outgrowth, presenting the storage space reduction when using the proposed approach by a VCS. Here we found that the repository size increases approximately 10 times faster when our approach is not used;
- How the repository grows when any image processing is performed, showing that repositories composed by images that suffered global transforms are reduced by 14 times when applying imaging processing techniques; and
- The performance contrasting image processing using GPU and CPU, where a speedup up to  $55 \times$  was found in relation to CPU.

For video artifacts, we evaluated:

- The size of delta storage contrasted to storing the whole video, where a reduction about 99.57% was achieved; and
- The performance contrasting video processing using GPU and CPU, where a speedup up to  $2.55 \times$  was found in relation to CPU.

Finally, for repository exploration we evaluated:

- The identification of dependencies among artifacts considering only support (widely used in the literature) against considering support, confidence, and lift, where we found approximately 23% of divergence;
- The identification of expertise, contrasting the use of coarse grain against fine grain analysis, presenting a 25% of deviation;
- The identification of expertise over time, where we could observe that expertise significantly varies during the project life, leading to imprecisions when considering the repository as a whole;
- The performance of our GPU implementation, showing a speedup up to 49.67 in relation to CPU; and
- The usability of our proposed approach, showing a success rate of 86.11% when participants were assigned to answer questions over a repository.

## 1.3 Organization

This thesis is organized in other five chapters, besides this introduction. Chapter 2 presents an introduction to Version Control Systems (VCS), where the most important concepts for a better understanding of the subsequent chapters are detailed, such as *diff*, *patch*, and *merge* operations. Additionally, the paradigm of centralized and distributed VCS are explained in this chapter.

In Chapter 3, the approach used to process image artifact over VCS, named *IMUFF*, is presented. This chapter begins presenting a real example regarding the importance of version control operations over this kind of artifacts. It then follows by an introduction to digital image and its underling structure. In the sequence, the specialized *diff*, *patch*, and *merge* operations are detailed. Additionally, the image processing techniques used for extracting an image transform and the underling data structure used to store the *delta* produced by *IMUFF* is presented. It then finishes by presenting the evaluation in relation to repository outgrowth and performance of our approach. Finally, this chapter also contrasts the related work with our approach.

Chapter 4 presents the approach for processing video artifact, called *VIMUFF*. Initially, the most important concepts in digital videos are introduced. In the sequence, the specialized *diff*, *patch*, and *merge* operations are detailed in addition to the underling data structure. The approach used for detecting modifications on videos is also carefully

detailed in this chapter. This chapter finishes by presenting an evaluation regarding to storage space consumption and processing time. At the end, we compare and contrast the proposed approach with related work.

Chapter 5 presents *Dominoes*, our GPU tool for extracting and analyzing artifacts over software repositories. It starts by presenting the underlying architecture used by Dominoes as well as the Dominoes pieces concepts and the available specialized operations. It then follows by presenting two usage scenarios of Dominoes: the calculation of dependencies among artifacts and the expertise identification, showing that direction matters for the former and that granularity and time matter for the latter. In the sequence, the evaluation is performed over Derby Apache project, a long living project. This evaluation is performed regarding dependency calculation, expertise identification, user evaluation, and performance. Additionally, we perform usability evaluation over Dominoes. In this chapter, related work is compared against Dominoes approach.

Finally, on Chapter 6 we present the conclusion of this thesis, showing the contributions regarding binary artifacts processing and data repository analysis. Additionally, we present some limitations that we found in the implementation of our approaches and enumerate future works to improve the work done so far.

# Chapter 2

## Version Control Systems

### 2.1 Introduction

Configuration Management (CM) has been developed since the 50's due to a necessity to control modifications over warplane and spaceship specifications [73, 39, 54]. After a while, between the 60's and 70's, CM also started to comprise software artifacts along with previously hardware artifacts. At this time, Software Configuration Management (SCM) [53] has emerged to deal specifically with software artifacts.

Many SCM definitions have been emerged, although the most accepted one defines SCM as “a discipline that applies technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a *configuration item*, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements” [1]. It is possible to state that SCM is not responsible to define how software engineering activities must be achieved, but to aid and keep up with how activities are made, being considered a supporting discipline to the whole process. According to Estublier [38], a typical SCM tries to provide services in the following areas:

- **Managing a repository of components:** it is necessary to safely store different components of a software product and all their versions. This topic includes version management, product modeling, and complex object management.
- **Help engineers in their usual activities:** software engineering involves applying tools over artifacts. As a consequence, SCM are expected to provide the correct artifact version as long as its location to the engineer. Typically this functionality is known as workspace control.

- **Process control and support:** in the 80's, it became clear that one of the biggest problems in relation to software development was related to people.

SCM can be used under different perspectives, according to the participant role in the software development process [7]. In this work, we are focused in the development perspective, where Version Control Systems (VCS) are a key element.

In this chapter, we provide an introduction to SCM, mainly focused on VCS. Section 2.2 introduces VCS basic concepts. Section 2.3 discusses the topologies employed by VCS. Section 2.4 presents the base operations of VCS. Finally, Section 2.5 presents some final considerations.

## 2.2 Basic concepts

Version control systems date back to the 70's and deal with the identification and evolution of software artifacts. Artifact in the VCS context represents anything that can be put under version control, such as files and directories in file-based systems [27]. SCCS [15] and RCS [101] were the first VCS tools to emerge. Since then, VCS evolved substantially, providing new functionalities and topologies.

Feiler [43] defines four types of VCS models used in a business environment: (1) The checkout/checkin model, focusing mostly on the versioning of product components as well as low level primitive operations to deal with those components; (2) The composition model, which allows a user to select which version of each artifact should build a given configuration; (3) The long transaction model, where users work on specific versions and use transactions for changes; and (4) The change set model, which supports changed-oriented configuration management and allows the user to visualize a configuration in terms of a collection of logical changes that can be linked to a change request.

In order to deal with artifacts, a VCS uses the concept of *version*. A version  $v$  represents a state of an artifact in a certain moment of time. According to Conradi and Westfechtel [27], depending on the intent of the version, it can be classified as *revision*, *variant*, or *cooperation*. A revision is made to supersede its predecessor and in this case, an artifact evolves as successive revisions. Variants are versions intended to coexist. Finally, versions can be maintained by each user to allow cooperation.

Most of the existing VCS work over files and directories in the file system. As presented in Figure 2.1, these files can be classified as *textual* and *binary*. Textual files are composed



by a collection of lines while binary files are opaque data.

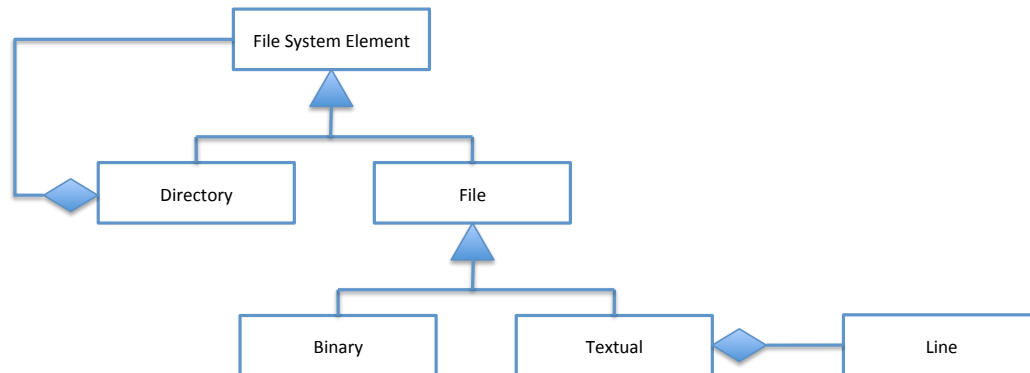


Figure 2.1: Elements versioned by a VCS [86].

The *repository* is responsible to store all artifacts that resides under VCS control as well as each artifact history (versions). Additionally, the history maintains the reason for a change, who made that change and when the change was made [33]. When a user desires to make a modification over an artifact, it must first be copied to the user's *workspace*. Changes performed over artifacts in this workspace are not immediately reflected to the repository, but can be submitted to the repository later on. By using workspaces, users can concurrently work on the same project in parallel.

In order to avoid artifact inconsistencies when submitted to a repository, a VCS employs a *concurrency control policy*, that can be divided into *pessimistic* and *optimistic* cases. When using a pessimistic concurrency control policy, a user can only modify artifacts that are not being modified by any other user by adopting a lock mechanism. On the other hand, by adopting an optimistic concurrency control policy, artifacts can be modified in parallel and these modifications can be combined (merged) later, when submitted to the repository, in order to produce the final version.

For situations where various users modify artifacts from the same project in parallel, conciliating these modifications can induce *conflicts*. These conflicts can be physical and logical. Physical conflicts arise when modifications are made in parallel at the same part of an artifact. In this case, the VCS cannot combine them automatically, requiring manual user intervention to solve these conflicts in order to put the repository in a consistent state. On the other hand, logical conflicts occur when modifications are made in parallel into distinct parts of the artifact, but after the automatic combination the artifact is put at an inconsistent state.

In order to postpone combining modifications made in parallel, a *branch* can be used. A branch is a new independent line of development that represents a collection of revisions

that must be kept separately from the main line of development. Normally a branch is often used to implement a feature or bug fix. Later on, a branch can be combined into the main line of development in order to reflect the modifications over the artifacts. Figure 2.2 illustrates the concept of a branch, where each node is a revision of its parent version (showed by the arrow) and the box shows the name of the branch. In this case, the branch that represents the main line of development receives the name of *master*.

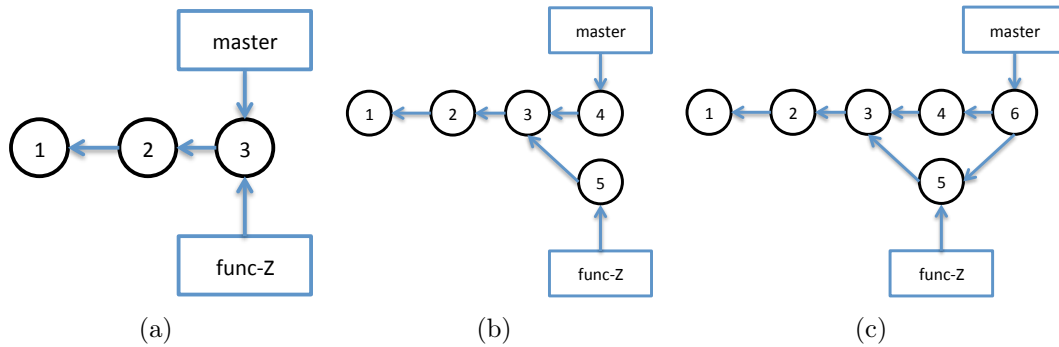


Figure 2.2: A branch for implementing a new function in the project. In (a) a new branch is created (named *func-Z*) derived from the main line (named *master*). In (b) modifications were made in both *master* and *func-Z* branches. Finally, in (c) the branch is merged to the main line of development.

In Figure 2.2(a), in addition to *master* branch, a new one (named *func-Z*) is created for implementing a new functionality in the project. From this moment, every change performed on each branch is kept isolated from changes in the other branch, as presented in Figure 2.2(b), where modifications were made in both *func-Z* and *master* branches. The modifications produced version 4 for the *master* branch and version 5 for the *func-Z* branch. Finally, in Figure 2.2(c), the branch *func-Z* has been merged into the *master* branch.

Each node in Figure 2.2 represents a *commit*, which is defined as the state of all artifacts under VCS in a specific time moment. In addition, a commit has the information of the author of the commit as well as the date it was performed and a message. At the end, the history of a VCS is composed by a collection of commits, as showed in Figure 2.3.

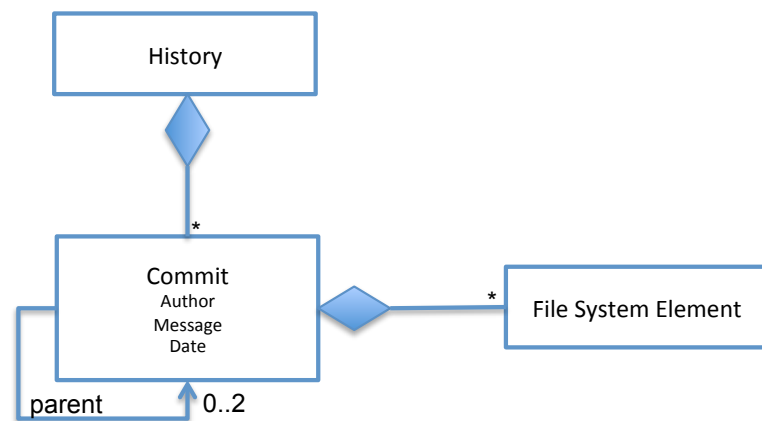


Figure 2.3: VCS history composition (adapted from [86]).

## 2.3 VCS topology

A VCS needs to define how the data is organized in a repository as well as its location and topology. This aspect is closely related to how developers use and collaborate in a project that employs configuration management techniques using a VCS. The collaboration aspect is closely related to the topology used by a VCS, being classified as *centralized*, or CVCS (Figure 2.4(a)) and *distributed*, or DVCS (Figure 2.4(b)).

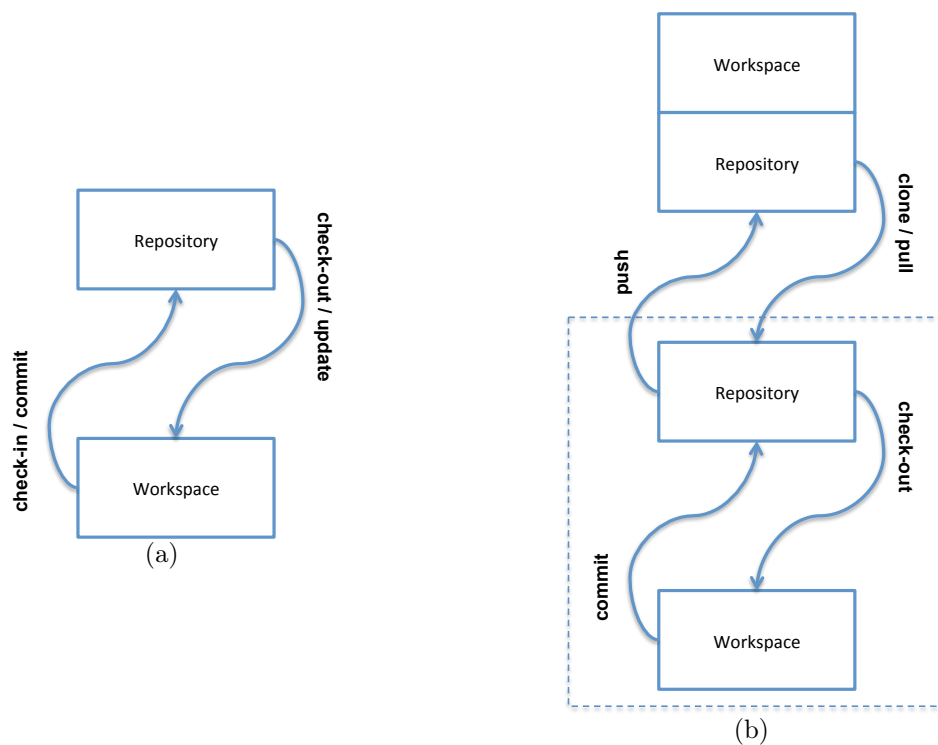


Figure 2.4: CVCS (a) vs DVCS (b) topology [85].

### 2.3.1 Centralized topology

CVCS relies on a central repository stored on a server, according to Figure 2.4(a). This model is adopted by VCS such as CVS<sup>1</sup> and Subversion<sup>2</sup>. In this case, when a user desires to work on a specific version of an artifact, it must be first copied to her workspace. The *check-out* command is responsible for copying the selected version to the user's workspace. Later on, after performing the desired modifications to the artifact, a *check-in* (also known as *commit*) command is responsible to submit these modifications to the repository. The collection of changes introduced by a commit is known as *changeset*. This guarantees that all modifications located at the user's workspace become available in the repository. Finally, the *update* command is used to bring parallel changes from the repository to the user's workspace. It is important to state that the *update* command automatically merges modifications into the workspace, requiring user intervention in case of conflicts.

For a long time, this was a standard topology employed by almost all commercial and open-source VCS. One advantage of the centralized topology is the level of control over the repository, allowing fine-tuning the desired access-control policy. Besides that, centralized topology can employ a pessimistic concurrency control policy for managing concurrency. However, there are some disadvantages: considering that a repository is located on a server, users cannot collaborate or perform a *check-in* or *check-out* during a communication shortage. This fact can preclude users from concluding their tasks, lowering their productivity. Additionally, the lack of offline commands, unavailability of the repository, among others [89], are considered negative aspects of a centralized topology.

### 2.3.2 Distributed topology

In order to minimize the negative aspects found on the centralized topology, the DVCS was developed (Figure 2.4(b)), steadily replacing the former generation of VCS. In this topology, used by Git<sup>3</sup> and Mercurial<sup>4</sup>, users do not simply perform check-out in order to retrieve the last version of artifacts found in the repository. Instead, the repository as a whole is retrieved, comprising all history of the project. It is important to state that *check-in* and *check-out* commands work exactly in the same way as discussed for the centralized model, however acting on the local repository.

---

<sup>1</sup>CVS website: <http://cvs.nongnu.org>

<sup>2</sup>Subversion website: <http://subversion.tigris.org>

<sup>3</sup>Git website: <http://git-scm.com>

<sup>4</sup>Mercurial website: <http://mercurial.selenic.com>

The interaction with the distributed repository is made by additional commands. According to Figure 2.4(b), the commands *clone*, *push*, and *pull* have been added to DVCS. The *clone* command – typically the first command to begin contributing to a project in a DVCS – transfers the whole remote repository to the user’s computer. The *push* command is used for sending the user’s modifications performed over her repository to a remote repository. On the other hand, for transfer changes made in parallel in a remote repository into a local repository, the *pull* command is used.

One advantage of using DVCS is the possibility of having any repository acting like a server for other repositories. This allows the definition of a hierarchy, where a repository is at the same time client of another repository and server to some other repositories.

## 2.4 Base VCS operations

VCS commands can be decomposed into a sequence of operations called in this thesis as *base operations*, presented in the vast majority of VCS. The VCS base operations are: *diff*, *patch*, and *merge*.

When performing a *check-in* command, the modifications performed by the user are sent to a repository. In a CVCS, the repository is located in another machine. In this case, in order to reduce the data that needs to be transferred, just the portion that has changed is submitted to the remote repository. This set of changes receives the name of *delta*. The process of finding this difference is made by using the *diff* operation. The *diff* operation is a common delta-calculus algorithm to detect changes between two versions of an artifact [61]. It is worth to mention that the *diff* operation is also used to reduce storage space requirements and provide understandable comprehension of changes. This operation is further detailed on Section 2.4.1.

After difference extraction by the *diff* operation, it becomes necessary to apply it to the artifacts located in the remote repository. In order to accomplish this task, the *patch* operation is used, further detailed in Section 2.4.2.

Finally the *merge* operation is performed in the VCS to reconcile two revisions created in parallel through an optimistic concurrency control policy. This reconciling may occur either (1) when someone wants to reintegrate a branch in the main development line or even in another branch; or (2) when someone wants to commit artifacts to the repository and it has already been changed in the meantime. The *merge* operation is detailed on Section 2.4.3.

### 2.4.1 *Diff*

Over 40 years of existence, VCS decisions have been influenced by storage space and processing time. According to Estublier [37], each revision of an artifact is 98% statistically similar with the previous revision. For this reason, storing just the difference between revisions reduces the storage space requirements and network bandwidth for data transferring, especially when a DVCS is used, which demands copying the whole repository. Additionally, the ability to visualize the difference between two versions of an artifact can benefit the comprehension of modifications.

As previously mentioned, most of the existing VCS work over files and directories. Those kinds of VCS are format agnostic, usually identifying the differences, or *delta*, between two artifact revisions in terms of the lines that were removed or added. This VCS strategy considers files and directories as the units of versioning and lines as the unit of comparison [84]. The line-based approach for finding a *delta* remains a very useful technique due to its efficiency, scalability, and generality [81].

Extracting the *delta* between two versions of an artifact is accomplished by the *diff* operation [60, 59], according to Equation 2.1:

$$\mathit{delta}_{i \rightarrow j} = \mathit{diff}(V_i, V_j), \quad (2.1)$$

where  $V_j$  and  $V_i$  represent two versions of an artifact and  $\mathit{delta}_{i \rightarrow j}$  is the difference between these two versions.

However, some kind of artifacts cannot be represented as a collection of lines, such as image and video, being considered as binary by the VCS. Binary artifacts are opaque data without any kind of processing performed over them by the VCS. As a consequence, the artifact is stored as a whole at the repository for each revision, independently of the amount and size of the content that has been modified. Besides consuming more space in the repository and requiring more network bandwidth to be transferred, it is almost impossible to comprehend the modifications performed in this kind of artifact.

As an alternative to this problem *VBinDiff*<sup>5</sup> can be used to extract the difference between two binary files in order to reduce storage and network bandwidth. However, instead of a set of lines added and/or removed, the difference is presented in hexadecimal, without helping the comprehension of such modifications. For example, using *VBinDiff* over two versions of an image artifact will not provide a better way to visualize these

---

<sup>5</sup>VBinDiff website: <http://www.cjmweb.net/vbindiff>.

differences.

In addition, artifacts that possess a well-defined syntax, such as XML, do not provide relevant information about modifications when treated as a simple set of lines. In this case, although it is possible to produce a *delta* between two versions, this *delta* is not able to provide the better comprehension about this modification.

Regarding to storage and reconstruction of textual artifacts, a VCS can use different strategies (see Figure 2.5):

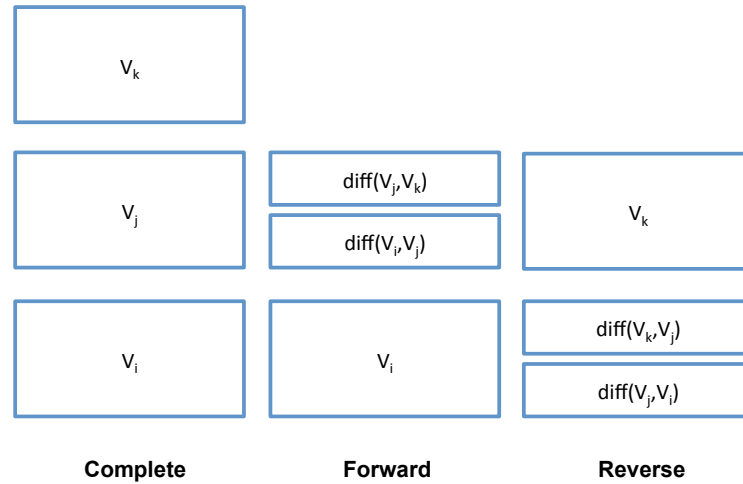


Figure 2.5: *Delta* storage strategies: Complete, forward, and reverse [85].

- **Complete versioning:** stores the whole artifact on each revision, independently of the amount of changes performed on it. Normally, this approach is used for binary artifacts due to the impossibility of considering their data as a collection of lines.
- **Forward versioning:** in this approach, the first version of an artifact is stored as a whole. For each subsequent version of the artifact, the *delta* between the prior and current version is stored. Reconstructing a specific version of an artifact consists of applying a set of *deltas* since the first version to the one being requested.
- **Reverse versioning:** this approach is exactly the opposite of forward. In this case, the most recent version of an artifact is stored as a whole. On each *commit* of a newer version ( $n$  for instance), the VCS calculates the difference between versions  $n$  and  $n - 1$  as a *delta*, storing  $n$  as a whole and the *delta* to  $n - 1$ . This approach is based on the fact that the vast majority of *check-outs* are on the last version of an artifact, which can be provided without any processing.

Additionally to the *diff* operation, the *diff3* operation is frequently used by VCS. Instead of considering just two versions of an artifact and extracting the difference between them, the *diff3* operation uses three versions of an artifact to process the difference. One version is considered the base *A*, while the other two, *B* and *C*, are considered modifications over *A*. By considering a common ancestor for processing the difference, a more precisely result is obtained. Normally *diff3* operation is used for conciliating parallel modifications made to an artifact from a common ancestor.

Figure 2.6 shows the *diff3* operation. When using *A* as the base version, it is possible to see that a modification has been made in the first line of version *C* while on version *B* there is no modification presented on this line.

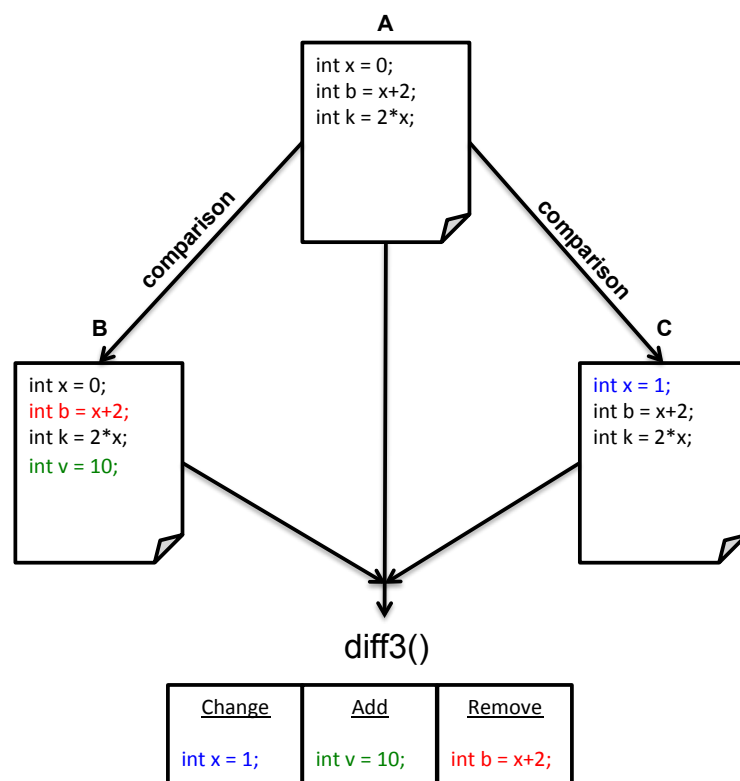


Figure 2.6: The *diff3* operation used to identify changes on parallel development.

As presented in this section, the necessity of specialized *diff* operation to deal with artifacts that cannot be treated as sets of text lines is clear. A specialized *diff* operation allows comprehension of performed modifications to an artifact based on its type. Besides that, it allows storing just the *delta* between revisions of an artifact, reducing the space required as well as the network bandwidth used to transfer these artifacts.



### 2.4.2 Patch

Sometimes it is necessary to retrieve an intermediate version of an artifact. In this case, supposing that a  $\text{delta}_{i \rightarrow j} = \text{diff}(V_i, V_j)$  was generated between two versions of an artifact,  $V_i$  and  $V_j$ , the *patch* operation is used to reconstruct  $V_j$  by applying  $\text{delta}_{i \rightarrow j}$  over  $V_i$ , according to Equation 2.2:

$$V_j = \text{patch}(v_i, \text{delta}_{i \rightarrow j}), \quad (2.2)$$

where  $V_i$  corresponds to a version of an artifact,  $\text{delta}_{i \rightarrow j}$  is the *delta*, and  $V_j$  is the requested version.

Due to the fact that *delta* generated by *diff* operation needs to be known by the *patch* operation in order to build a requested version of an artifact, it is normal to develop both operations together in order to support a specific artifact type.

### 2.4.3 Merge

The *merge* operation aims at conciliating variants created in parallel. To do so, given two versions of an artifact,  $B$  and  $C$  both descendent from version  $A$ , the *merge* operation generates a new version  $D$  of this artifact comprising modifications performed on both versions  $B$  and  $C$ , according to Figure 2.7.

As the *diff* operation, *merge* can be subdivided in *two-way* and *three-way* merge. The two-way merge considers two versions of a specific artifact and tries to combine their differences. In this case, when the same line differs in both variants, a conflict will be marked, requiring manual user intervention. On the other hand, the three-way merge considers the common ancestor of an artifact for combining two versions modified in parallel. When considering Figure 2.2, the three-way merge will use the version of artifacts presented in commit 3 when merging branch *func-Z* into *master*, as commit 3 is the common ancestor for both branches.

According to Babich [9], three fundamental problems in team coordination can be observed: (1) shared data, handled by VCS with separate workspaces; (2) simultaneous update, handled by VCS with a concurrency control policy; and (3) double maintenance, handled by the *merge* functionality of the VCS. From this standpoint, it is possible to see the importance of the *merge* operation when working in parallel. As VCS do not provide *merge* operations for binary artifacts, there is no support for combining parallel changes,

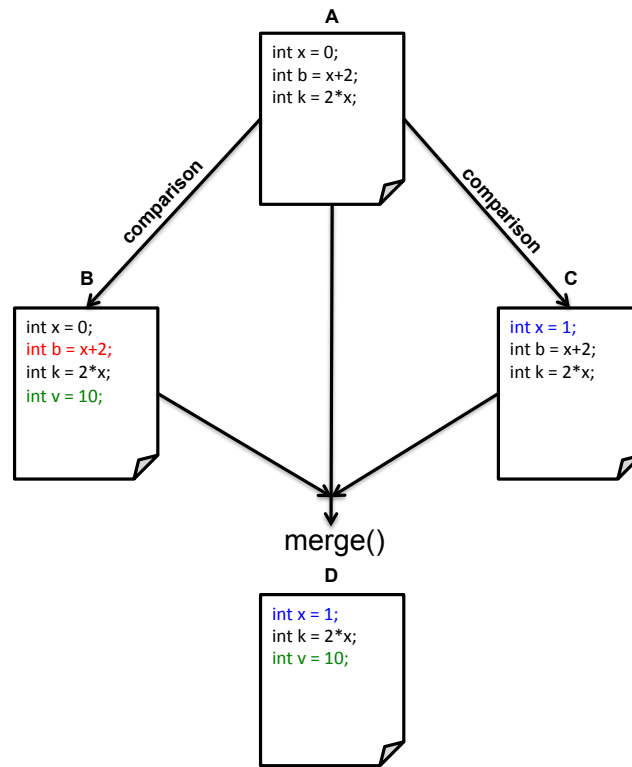


Figure 2.7: The *merge* operation aims at conciliating modifications from variants *B* and *C*, generating version *D*.

and so the user is required to select (based on one's knowledge) which version to maintain when a conflict occur.

The importance of the *merge* operation is closely related to the adopted concurrency control policy [95]. By adopting a pessimistic concurrency control policy, the *merge* operation is not necessary as the artifact is changed by just one user at a time. However it usually becomes counterproductive when the number of users performing parallel tasks increases. On the other hand, by adopting an optimistic concurrency control policy, the use of *merge* becomes necessary in order to conciliate parallel modifications performed on an artifact by different users.

## 2.5 Final considerations

Although VCS are considered sufficiently mature for practitioners, there is the necessity of further improvements. By considering artifacts as a collection of lines, VCS are able to process different types of artifacts in a reasonable amount of time. However, this approach presents some disadvantages. The first one is the impossibility to present the difference between versions of an artifact in a more adequate way for structured textual

artifacts, such as XML. Additionally, when a VCS cannot treat an artifact as a collection of lines, it is considered as binary and each version is stored as a whole. For such cases, no difference is calculated, requiring high storage space. Indeed, conciliating changes performed in parallel to binary artifacts is almost impossible. When considering DVCS topology, bandwidth becomes an issue since the whole repository is transferred from and to the users.

According to Estublier [38], the research and development performed on VCS can be considered independent of any programming language and any semantic aspect of an artifact. Thus, many weaknesses of VCS come from having little knowledge about the artifact. To overcome this issue it is necessary to increase VCS power by providing knowledge on some artifact types. However, this must be made without having to pay too much in terms of complexity, efficiency, and generality to achieve this objective.

As a consequence, it becomes important to process each kind of artifact using specialized algorithms. In this context, Mens [81] suggests the use of a more reliable *merge* operation considering the internal structure of an artifact. As *merge* operation usually relies on both *diff* and *patch*, this suggestion also extends to the aforementioned operations.

On the other hand, performing exploratory analysis does not require just extracting commits' metadata on the VCS history. In fact, it also requires extracting the data itself for these different versions presented in the repository in order to calculate dependency among artifacts, modified methods, expertise and so on. Doing this kind of analysis requires a high processing effort, spending too much time depending on the size of the project and its lifetime. This fact leads to scoping the data to allow it to be processed in a feasible amount of time. In this case, off-line specific analysis can be done with the scoped data.

# Chapter 3

## Image-aware version control

### 3.1 Introduction

In many development areas, the preservation of intermediate revisions of evolving artifacts is very important to allow rollback operations in the case of an undesired change or simply to check modifications between versions of the software. In the videogame industry, this is particularly relevant as at the end of the development stage one often needs to optimize and streamline performance by reducing and optimizing the artifacts.

While using lines as the unit of comparison may be suitable for text-based artifacts, some artifact types, such as images and videos, heavily used in the game industry, do not correctly fit into this data model due to their binary nature. For instance, Git stores different revisions of a complete new binary artifacts as individual files without any *delta* information related to their predecessors, making it difficult to discern the modifications between revisions [59] and leading to large repositories and considerable transfer times.

In this chapter, we present *IMage UFF* (IMUFF)<sup>1</sup> [32, 31], which considers images as first-class elements in the context of version control. *IMUFF* uses the change set model to represent the set of modifications that forms a logical image change, allowing a more accurate merge [19] of two modified images based on the same ancestor. Our approach customizes the VCS base operations (i.e., *diff*, *patch*, and *merge*) to deal with image artifacts. We allow people to deal with the parallel maintenance of images by providing adequate *merge* support. Also, our contextual *delta* has two main purposes: allow image change identification for the end user and avoid high storage space requirements and bandwidth consumption. In order to reduce the processing time, our approach employs a

---

<sup>1</sup>UFF stands for the name of the author's university.

parallel architecture based on a GPU to implement *diff*, *patch*, and *merge* operations on images.

Section 3.2 starts the discussion showing a motivational example for the problem of managing multimedia artifacts by presenting the case study for a real videogame. Section 3.3 introduces the concept of digital images while Section 3.4 discusses the proposed approach for the solution. Section 3.5 presents the evaluation of the approach and Section 3.6 discusses some threats that can affect the validity of our evaluation. Finally, Section 3.7 presents related work and Section 3.8 presents the final considerations of this chapter.

## 3.2 Motivational example: JECRIPE project

We used the JECRIPE <sup>2</sup> [18] game to better illustrate the problems related to the adoption of a common VCS for projects that are heavily based on multimedia artifacts. At first, each artifact is classified as sound, image, 3D model, source code, or video.

Figure 3.1(a) shows the total number of artifacts and their corresponding percentages, for each type of artifact. It can be seen that JECRIPE uses more multimedia assets than source code artifacts.

Figure 3.1(b) shows the amount of space and percentage required by each type of artifact. It is possible to see that image artifacts require more storage space than any other type of artifact. An important relation shown here is the storage space needed for source code artifacts. Although a reasonable number of source code artifacts can be found in this project, they occupy less than 1 MB. Their storage space requirements have little impact on the repository size.

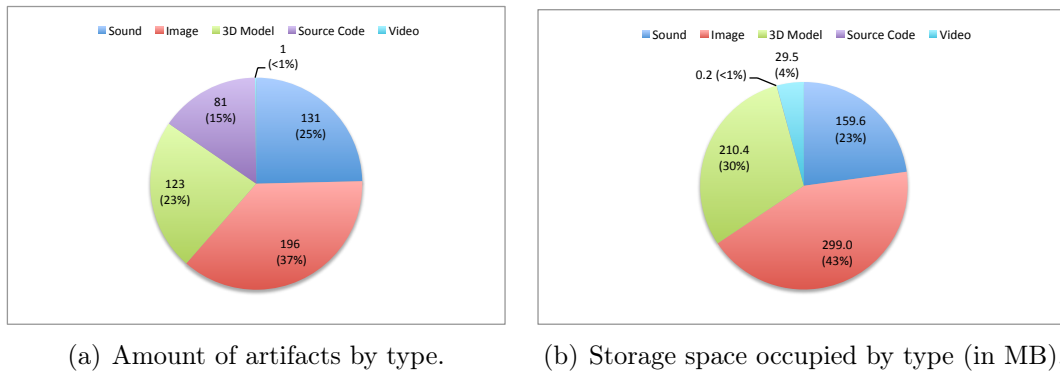


Figure 3.1: Study of different artifact types.

Although there are guidelines and descriptions of best practices to deal with multime-

<sup>2</sup>JECRIPE is a game whose target are children with Down's syndrome.

dia artifacts [8, 63], the absence of tailored support can produce side effects. Sometimes, a high volume of information can be recorded from user actions on an image editing software. The visualization of such user action histories can be used to distinguish modifications between two versions of an image, and has become a popular topic of research [70, 51, 67, 55]. Unfortunately, this approach relies on knowing which feature and tool has been used to edit the image artifact, which makes integration with a generic VCS difficult.

### 3.3 Digital image

In digital imaging, images are described by a matrix of numerical values, each value describing a single point in a raster image, called pixel. A pixel is the smallest addressable element in a display device that can be represented or controlled. A pixel is made of three components, also known as channels: *red* (**R**), *green* (**G**) and *blue* (**B**). For some kinds of images, an extra channel is also used, called *alpha* (**A**), which is responsible to store the transparency of the pixel in order to perform different effects such as alpha blending, as illustrated in Figure 3.2.

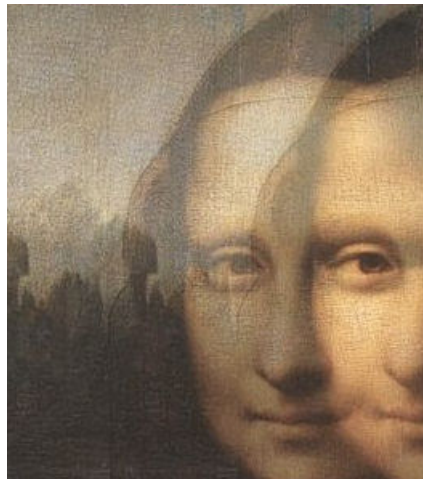


Figure 3.2: Mona Lisa using an alpha blending effect.

The collection of pixels of an image can be represented by a matrix, where rows represent the  $y$  coordinate and columns the  $x$  coordinate. Within each entry of  $(x,y)$  coordinate, there is a pixel, that can be represented as a vector of three (R,G,B) or four (R,G,B,A) elements.

In order to represent a color, normally each channel uses a byte, ranging from 0 to 255, which gives 256 distinct tones for each color. This approach makes possible to show up to

$256^3$  different color values by mixing these three channels together. As an example, Table 3.1 shows a matrix representation of a  $3 \times 3$  pixel image that has a white dot in its center. It is important to notice that for some images that use the alpha channel, the number of colors stays the same, being this extra channel used only for blending operations.

Table 3.1: Matrix of a  $3 \times 3$  pixels image with a white dot in its center.

$\langle 0, 0, 0 \rangle$	$\langle 0, 0, 0 \rangle$	$\langle 0, 0, 0 \rangle$
$\langle 0, 0, 0 \rangle$	$\langle 255, 255, 255 \rangle$	$\langle 0, 0, 0 \rangle$
$\langle 0, 0, 0 \rangle$	$\langle 0, 0, 0 \rangle$	$\langle 0, 0, 0 \rangle$

According to how image files are represented, performing version control operations in this kind of artifact requires processing all these pixel elements. Additionally, revision data information has to be stored in such a way that allows fast backward and forward reconstructions.

Although it is possible to find different kinds of image layouts, such as multi-layer, mipmaps, or vector images, our focus is on raster-based images. Approaches that deal with high level images are *operation based*, normally tied to a specific tool [26]. These approaches record the history of all operations performed by each user, enabling undoing them if necessary. However, aiming at avoiding tool dependency, we opted for a *state based* version control. This allows us to be compatible with popular VCS such as Git and Subversion. Apart from that, layered and vectorial images are not supported by the vast majority of popular image extensions (e.g., PNG, JPG, and TGA) and in general cannot be directly used in a project without their proper conversion to a bitmap layout.

## 3.4 Proposed approach

As discussed before, artifacts treated as binary by VCS require specialized operations of *diff*, *patch*, and *merge* to be processed. This section presents the approach used for specializing these operations in the image context, providing insights about their integration into VCS.

Besides that, our approach applies image-processing techniques in order to extract high-level transformations from images, such as translation and rotation, presenting to the user a more semantic view of his/her modifications across different versions of an image artifact. A data structure was conceived and employed by *IMUFF* to accommodate all data used during image processing and versioning.

### 3.4.1 The *diff* operation

The extraction of difference is not well defined for image artifacts. Some approaches are based on general image comparison visualization: the popular ones include side-by-side comparison, layer-based differences, image overlay [16], flickering difference regions, and linear and non-linear operations on images [26]. For better results, this process usually requires finding pixels that differ in two given images. Instead of transferring the responsibility of finding modifications in images to the end-user through visual comparison, our approach can pinpoint the exact location where a modification happened in the image.

This operation in *IMUFF* is carried out by locating the difference between two images and saving its result as another image, which is called the *delta* between them. We wish to give this *delta* contextual information to provide an adequate way to locate the difference by just observing the resulting image. In other words, this *delta* is not only designed to reduce space consumption and improve bandwidth, but also to provide a fast visual identification of the changes between two images. As an example, a *diff* operation on images *A* and *B* produces image *C* as a delta, where the pixels that are equal in both images (*A* and *B*) correspond to black pixels in *C*, and the pixels that differ appear with a different color in *C*.

The *diff* operation is made by using a binary **XOR** ( $\oplus$ ) over each channel for all the pixels of both images *A* and *B*. We chose this operator due to its property of running in parallel over the GPU in a single instruction as well as its good pixel change representation, allowing bidirectional use to reconstruct either version. As an example, the *diff* of a binary RGB of green (R=00000000; G=11111111; B=00000000) and blue (R=00000000; G=00000000; B=11111111) pixels would lead to a cyan pixel (R=00000000; G=11111111; B=11111111) after applying the **XOR** operation.

In our proposal, we executed the *delta* computation using the GPU architecture. In this case, we process each pixel independently and concurrently from each other. This modification produces Algorithm 1, where *thX* and *thY* are variables automatically calculated by the built-in CUDA context mechanism used to locate the specific data element that a thread will operate on.

To better contextualize this operation for image artifacts, we used two images of the “Where’s Wally?” game <sup>3</sup> (also known in the US and Canada as “Where’s Waldo?”). In the first image, there is a scene where Wally is absent, while in the second image he is

---

<sup>3</sup>Image used from <https://www.usbio.net/misc/newsletterseptember2011>.



**Input:** Image A, Image B

**Output:** Image C

```

1
  C.At(thX,thY).R = A.At(thX,thY).R  $\oplus$  B.At(thX,thY).R;
2
  C.At(thX,thY).G = A.At(thX,thY).G  $\oplus$  B.At(thX,thY).G;
3
  C.At(thX,thY).B = A.At(thX,thY).B  $\oplus$  B.At(thX,thY).B;
4
  C.At(thX,thY).A = A.At(thX,thY).A  $\oplus$  B.At(thX,thY).A;

```

**Algorithm 1:** Algorithm used for image *delta* generation in GPU.

somewhere in the scene. The *diff* operation receives both images and produces a *delta*, showing the exact location of Wally. The result of this operation is shown in Figure 3.3. The leftmost image is the scene without Wally, the center one is the image where Wally is found, with the one in the right is the *delta* image, with pixels different than black for unmatched colors in the same (x,y) position for both images.

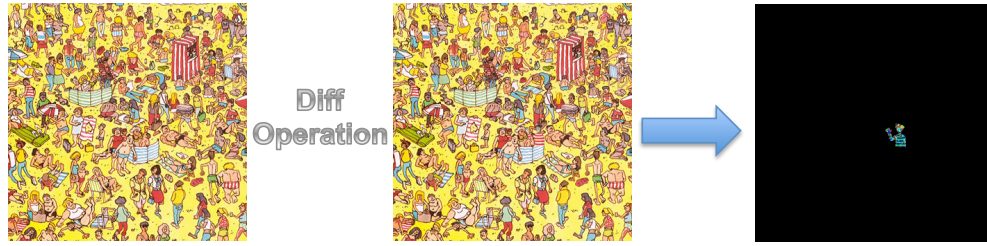


Figure 3.3: Applying a *diff* operation to images to obtain their *delta*. In the leftmost image a scene without Wally. In the center one, the same scene with Wally, and on the right image, the *delta* between them.

As can be seen, most of our *delta* image consists of black color (zeros), which leads to a small size after a lossless compression (using a PNG format), requiring less storage space and network bandwidth during commit and check-out. We should point out that usually a small *delta* between two consecutive versions is expected [37].

### 3.4.2 The *Patch* operation

Another important operation in VCS is the application of *deltas* to an artifact, to reconstruct its older or newer versions. For image artifacts, this is also an important and necessary operation due to the frequent modifications in their lifetime.

As stated before, we use the binary **XOR** operator to identify the pixels that have different colors on any channel, producing an image that represents the *delta*. Using the *patch* operation, it is possible to reconstruct an artifact based on its *delta* and its previous or next version (indeed our *delta* can be used both forwards and backwards). As

an example, given images A and B, which correspond to two versions of the same image, and an image C that contains the *delta* generated by  $\text{diff}(A,B)$ , we can rebuild image A by applying image C to image B. On the other hand, we can also reconstruct image B by applying image C to image A. Using the same example from Section 3.4.1, it is possible to apply the **XOR** operator to blue (R=00000000; G=00000000; B=11111111) and cyan (R=00000000; G=11111111; B=11111111) to reach a green (R=00000000; G=11111111; B=00000000) pixel. Beside this specific example, the **XOR** operator can be applied to any color to reach the original one at that pixel position. Algorithm 2 shows the *patch* operation performed with a GPU architecture, which is the same as presented before for *diff*, but changing the parameters order.

**Input:** Image A, Image C

**Output:** Image B

```

1 B.At(thX,thY).R = A.At(thX,thY).R  $\oplus$  C.At(thX,thY).R;
2 B.At(thX,thY).G = A.At(thX,thY).G  $\oplus$  C.At(thX,thY).G;
3 B.At(thX,thY).B = A.At(thX,thY).B  $\oplus$  C.At(thX,thY).B;
4 B.At(thX,thY).A = A.At(thX,thY).A  $\oplus$  C.At(thX,thY).A;
```

**Algorithm 2:** Algorithm for GPU image patching.

Figure 3.4 shows the results of applying this algorithm to the same “Where’s Wally?” scene. The leftmost image shows a scene where “Wally” is not shown while the center one shows the *delta* we want to apply to get the final revision. The image on the right shows the result of this operation.

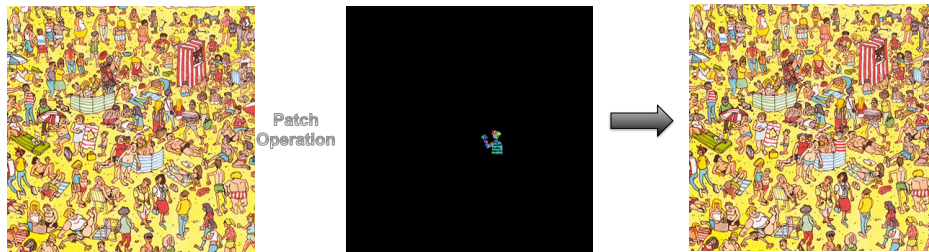


Figure 3.4: The left image shows a scene without Wally. In the center one, the *delta* from one revision of a scene where Wally is shown, and the one on the right has the reconstruction of the scene.

Using our approach we can recognize some important properties of the *patch* algorithm: identity, commutativity, and bidirectionality. Due to identity, patching an empty *delta* to an image produces an unchanged image (Equation 3.1). By commutativity, patching the *delta* to the image or the image to the *delta* produces the same image (Equation 3.2). Finally, by bidirectionality, patching the *delta* to any one of the original images

produces the other image (Equation 3.3). For all Equations 3.1, 3.2, and 3.3,  $A$ ,  $B$  and  $C$  are images of the same size.

$$\text{patch}(A, C) = A \text{ if } C \text{ is an empty } \textit{delta} \quad (3.1)$$

$$\text{patch}(A, C) = \text{patch}(C, A) = B \quad (3.2)$$

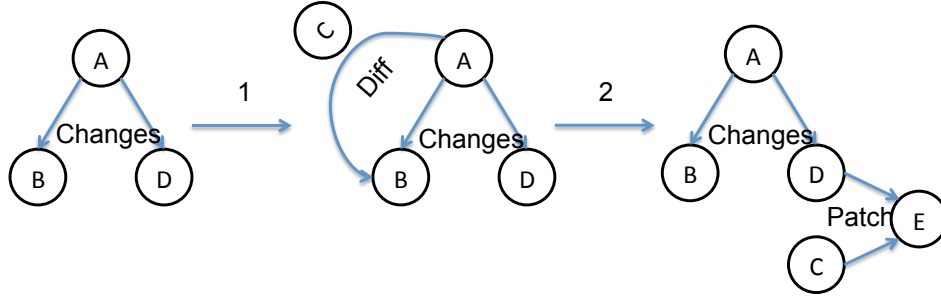
$$\text{patch}(A, C) = B \text{ and } \text{patch}(B, C) = A \quad (3.3)$$

According to Estublier [40], bidirectional *deltas* allow a more powerful version selection and at the same time provides better assistance to the user during the composition of a particular version with the use of the change-set model (the model used by *IMUFF*). The bidirectionality of the binary **XOR** operator allows *IMUFF* to obtain the original image  $A$  from a changed image  $B$  by simply applying the same *delta* to  $B$ , without any additional operation. For example, in Figure 3.4, the *delta* could be applied to the modified image (rightmost image), also using Algorithm 2, to get the prior revision state (leftmost image), thanks to the bidirectionality of the binary **XOR** operator.

### 3.4.3 The *Merge* operation

As said before, *IMUFF* uses the change set model [43] to represent the logical changes to images. In this case, given two images,  $B$  and  $D$ , both revisions of image  $A$ , the changes made to the  $B$  and  $D$  should be blended to get a final image  $E$ . *IMUFF* uses a three-way merge technique to allow a more precise combination among parallel development from a common ancestor. Our merge relies on using both the previously presented *diff* and *patch* operations in two steps, as in Figure 3.5. In the first step, images  $A$  and  $B$  are chosen (or  $A$  and  $D$  with no loss of generality) and a *diff* operation is applied to detect the changes between them, as shown by image  $C$ . In the second step, the changes encoded in image  $C$  are applied to image  $D$  (or image  $B$ , if  $A$  and  $D$  were the ones used in the *diff*) through a *patch* operation, to produce image  $E$ . Image  $E$  represents the merging of the changes introduced into images  $B$  and  $D$  in parallel. Algorithm 3 shows the whole process.

To better illustrate the *merge* operation, Figure 3.6 shows two examples where modifications were made in parallel to image  $A$  by different users, generating images  $B$  and  $D$ . In the first case (Figure 3.6(a)), both users merge their local copy, generating the final

Figure 3.5: Steps needed to execute *merge*.

**Input:** Image A, Image B, Image D

**Output:** Image E

1 Image C = Diff(A, B);

2 E = Patch(D, C);

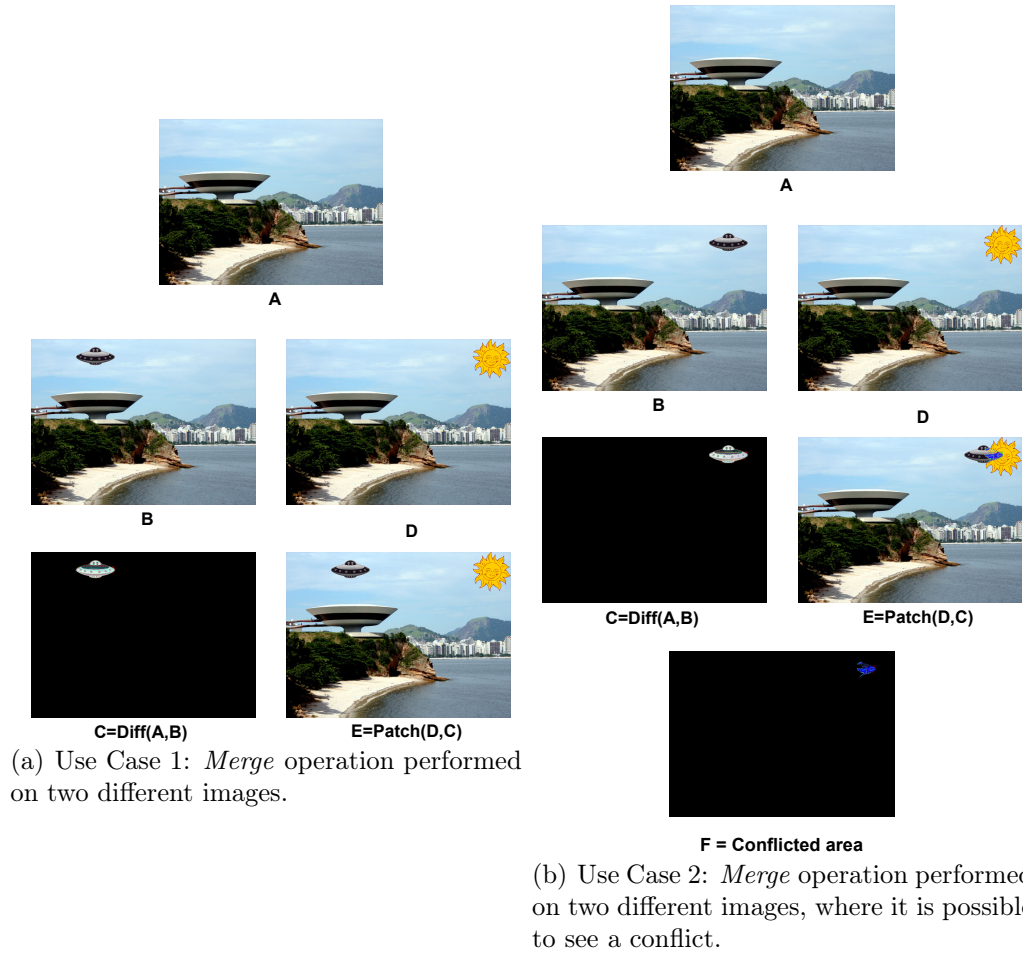
**Algorithm 3:** Algorithm used to execute a *merge* operation.

image E, which reconciles the two modifications made in parallel, without any physical conflict.

In the cases where the same image area is changed, a physical conflict is detected in the same way as with text-based VCS. In this sense, both methods (ours and the text-based approach) require user intervention to solve the conflict. This situation is illustrated in Figure 3.6(b), where pixels at the same location have been concurrently modified. *IMUFF* automatically detect and indicate whether a conflict occurred with the use of the **AND** ( $\otimes$ ) operator, facilitating conflict resolution during the integration of parallel development. For  $C = \text{diff}(A, B)$ ,  $C' = \text{diff}(A, D)$ , and  $F = C \otimes C'$ , a conflict is found when the sum of all pixel values in  $F$  leads to a value other than zero. In addition, the exact location of the conflicting pixels is shown on  $F$  wherever the pixels are other than black. In this case, for the situation shown in Figure 3.6(b), the  $F$  image would lead to all black pixels, except for the exact location where the sun touches the spaceship. This location would have pixels other than black, enabling the visualization of the conflicted region.

The above operations can be made in different directions to produce the same result. So, assuming that  $B$  and  $D$  are two revisions that descend from  $A$ ,  $\text{merge}(A, B, D)$  is the same as  $\text{patch}(D, \text{diff}(A, B))$  or  $\text{patch}(B, \text{diff}(A, D))$ , which in fact is the same as  $\text{merge}(A, D, B)$ . Using the change set model as well as the properties defined in Section 3.4.2 allows seamless data manipulation.

Finally, we should say that, as a toolkit, *IMUFF* cannot perform all the operations found in VCS *per se* (such as a *commit* or a *merge*). Instead, it is meant to be triggered

Figure 3.6: Applying a *merge* operation.

by a VCS, to efficiently deal with image artifacts. For the same reason, *IMUFF* is not responsible for deciding when the merge should happen (before or as part of the commit). The VCS designer should make this kind of decision.

### 3.4.4 Image Processing Techniques

According to Estublier [37], it is common to expect small changes between consecutive revisions. However, for image artifacts, global transformations can also be applied, such as translation, reflection, rotation, and bright adjustment. This kind of transformation leads to a *delta* that consists of all the pixels in the image, providing no hints on which kind of action has been done to the original image. This kind of *delta* provides no visual support to help the user to solve conflicts during a merge. Additionally, a *delta* produced by a global transformation usually requires more storage space than storing the whole modified file. This is a side-effect of compression algorithms, that usually work by taking into account the neighboring pixels that have similar colors, which are stored only once.

In the cases where a **XOR** has been applied to images that have been modified by a global transformation (such as a rotation), the number of pixels with the same color is very small, as shown in Figure 3.7. This happens mainly because during a global transformation, all the pixel colors in the modified image will not match the color from the original image, producing a color other than black (which is the color produced by applying **XOR** to pixels of the same color).

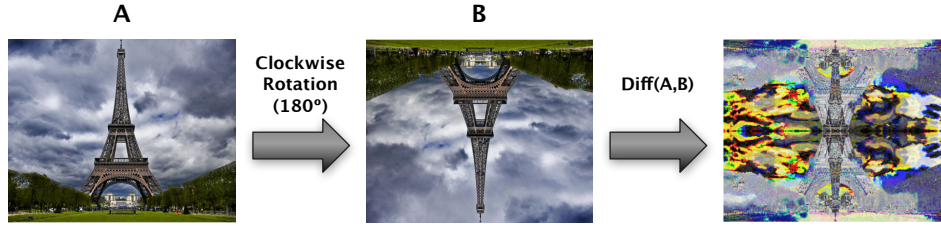


Figure 3.7: The *delta* generated after applying a global transformation to an image.

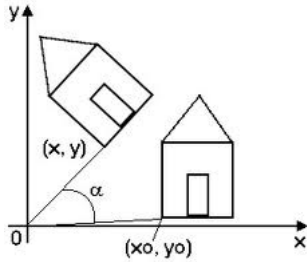
Amongst all the global transformations that can be applied to an image, a special set of them can be classified as rigid transformations. A rigid transformation is related to changing the position of a pixel in the image and can be defined by a bijective function for each pixel [48]. These functions are reversible, permitting the retrieval of the original or final position of the pixel with the application of this function. Rigid transformations can be detected by *IMUFF*, and include translations, rotations, and reflections. The reversion of rigid transformations can take place through a linear operation such as  $I_m(x, y) = U \times I_o(x, y)$ , where  $I_m$  is the final image,  $I_o$  is the original image, and  $U$  is a rigid transformation defined by a function. It is important to note that filter operations cannot be defined by a bijective function and, for this reason, are not considered here. For example, applying a blur filter would generate a *delta* that consists of all the pixels that are not black. Hu et al. [58] can detect this kind of transformation as well as many others but their approach requires at least three minutes to process an image of  $512 \times 512$  pixels in CPU, being not feasible for using it in a version control.

When looking at both the original (A) and the modified image (B) in Figure 3.7, it is easy to see that a  $180^\circ$  clockwise rotation has been performed. During a merge conflict, the user could solve the conflict by applying the operations in sequence, for example. However, understanding what has been done during a conflict resolution just by analyzing this *delta* is almost impracticable. Furthermore, storing  $U$  instead of the whole *delta* could lead to a smaller space consumption.

*IMUFF* uses image-processing techniques to identify a rigid transformation through the OpenCV [17] library. A rigid transformation is represented by a  $3 \times 3$  homogeneous



matrix composed by a rotation, a reflection, and a translation [2]. As an example, Figure 3.8 shows a homogeneous matrix used to rotate an image. In this case, a point  $(x_0, y_0)$  of the image is rotated by  $\alpha$  degrees (performed by multiplying a vector by a matrix), generating a new  $(x, y)$  rotated point. It is important to note that this operation should run for all pixels in the image in order to have the image rotated.



$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}$$

Figure 3.8: Homogeneous bi-dimensional rotational matrix applied to an image.

The extraction of the matrix  $U$  requires analyzing at least three independent well-known points (forming a triangle) in order to infer which transformation has been applied to the image [90]. The extraction of those points involves finding features that are preserved in both the original and the modified images, and is done using the SURF (Speeded Up Robust Features) algorithm [11]. A feature in this context represents a pixel that is very likely to be the same in the original and modified images, taking its neighborhood into consideration. From the set of detected features, the three best ones are picked and their pixel positions  $(\mathbf{x}, \mathbf{y})$  are used to build the triangle and recover the rigid transformation. As an example, Figure 3.9 shows the three best features detected to build up the  $U$  matrix. From this example, one can see that a  $90^\circ$  rotation has been done to the original image, leading to a rigid transformation that completely represents this operation.

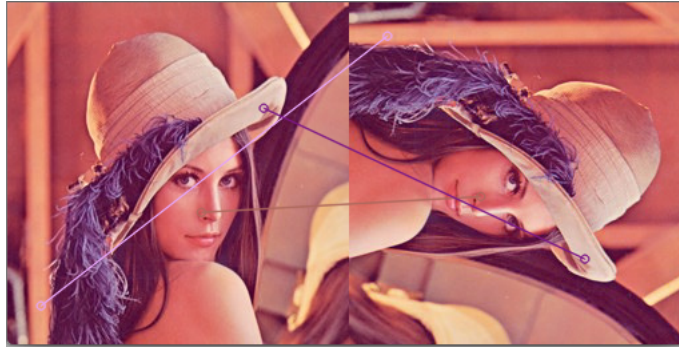


Figure 3.9: Mapping characteristics of images for extracting a rigid transformation.

As said before, translation, rotation, and reflection transformations can be detected

by *IMUFF*. For other transformations, the whole *delta* is stored in the recovery of a version, without taking the *U* matrix (set to an identity matrix) into account. Besides, due to floating point imprecision, some misaligned pixels can be seen in the original and reconstructed images when using *U*. To minimize this situation, we still save a *delta* between the two images as well as *U*. However, this *delta* is not the difference between the original and modified images (such as  $\text{diff}(A, B)$  in Figure 3.7) but the difference between  $I_m$  and the modified image ( $\text{diff}(I_m, B)$  in the former case). As  $I_m$  tends to be much more similar to the modified image than to its previous version, only those misaligned pixels are stored, leading to a very small *delta* when compared to the whole difference between  $I_m$  and the original image. In the end, reconstructing a version requires applying *U* to an image and then performing the **XOR** operation using this *delta*.

### 3.4.5 Data structure

Using a specialized approach to deal with specific artifact types allows significant improvements in *diff*, *patch*, and *merge* operations and data storage. The effort of the VCS to reduce the size of an artifact through compression is normally not effective for binary files. This is true for certain kinds of images, as they are already compressed and could be stored without any further processing. *IMUFF* takes advantage of this fact and only compresses images that have not already been compressed, such as bitmap files, avoiding redundant processing.

*IMUFF* uses a data structure where each image artifact has its own *database*, created when an image undergoes by modifications. This database consists of a history table and packages, as shown in Figure 3.10 for *Image A*. The figure shows that the database for *Image A* has two packages, ( **PCK-1** and **PCK-2**), each one responsible for storing a collection of *deltas*. Each package is mapped directly to a physical file, leading to two files in this specific case.

Each *delta* in the package represents a change set performed on an image, followed by its respective rigid transformation (when it is successfully found, otherwise it is an identity matrix). Figure 3.10 shows that **PCK-1** stores four *deltas* and their respective matrices (*M*, *B*, *C*, and *D*), while **PCK-2** stores three *deltas* as well as their respective matrices (*K*, *F*, and *G*). Each package can store up to a certain amount of data (configurable through a parameter). When this value is exceeded, a new package is automatically created to accommodate new *deltas*. The main motivation for storing more than one *delta* in the same package is to use less space, as the probability of having more colors of



the same value in the *deltas* packages tends to rise. Besides the likelihood of having more pixels with the same color as more *deltas* are included in the same package, we decided to split it. One motivation for this is to reduce data corruption. In the event of one of these packages becoming corrupted, the versions stored in the other packages can still be retrieved. Our second motivation was to improve the ability to search for a specific *delta*, as moving inside a big file tends to be a slower process. One important fact is that the first *delta* in package **PCK-2** only consists of one matrix ( $K$ ), meaning that an accurate transformation mapping between the two images involved has been found, and we can assume that the base image consists of only black colors.

The second component in this database is the history table, which is responsible for indicating how an older version of the artifact can be built. It stores a version navigation field (i.e., a source and target version), the package where one can find the right *delta* or change set, the offset (in bytes) for the beginning of the desired *delta*, and, finally, the size of the *delta* (considering the change set and matrix size, the latter being constant). In this case, reconstructing an older version requires locating the right *delta* row by querying the revision table and then accessing the right package and offset. As an example, and considering the *Image A* database in Figure 3.10, retrieving the prior revision of *Image A* ( $V_n - > V_{n-1}$ ) involves applying the *delta* located at the beginning of **PCK-1** (the offset is zero). On the other hand, retrieving version  $V_{n-2}$  from version  $V_{n-1}$  of *Image A* involves applying the *delta* located in package **PCK-1** and offsetting 12 bytes (the offset equals 12), which represents the second *delta* in this package (along with the matrix  $B$ ). As well as the package, the database, together with this table, is mapped to another physical file.

*IMUFF* employs a lossless image *delta* format (PNG) during the compression of each package that includes an image database. This kind of compression maintains the fidelity of the original image while reducing the used disk space. Actually, each image *delta* is compressed and then put together with images in the same package.

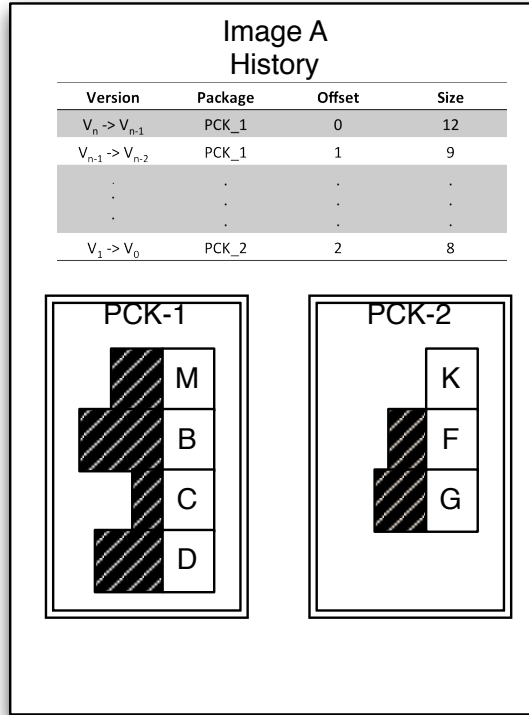


Figure 3.10: Data structure describing information stored for *Image A*. A table is used to locate revision data in different packages. Black hashed blocks represent *delta* size for each revision while white blocks store its matrix.

## 3.5 Evaluation

This section presents an evaluation of our approach in terms of repository outgrowth and performance. For all experiments, an Intel Core 2 Quad Q6600 2.40GHz PC with 4GB RAM and a nVidia GeForce GTX580 graphics card was used. This GPU has 16 Stream Multiprocessors with 32 Stream Processors each, giving up to 512 cores, and a market price around \$300,00 in the US in 2015. For the evaluation, we developed a GUI (Graphical User Interface) to activate the process and inspect the generated results, as presented in Figure 3.11. However, in real scenarios, a more automated usage, such as a pluggable module for VCS, should be used.

### 3.5.1 Repository outgrowth

In this section, we present a study to demonstrate and compare the repository outgrowth. Git, Subversion and Mercurial, some of the most popular VCS used nowadays, are used

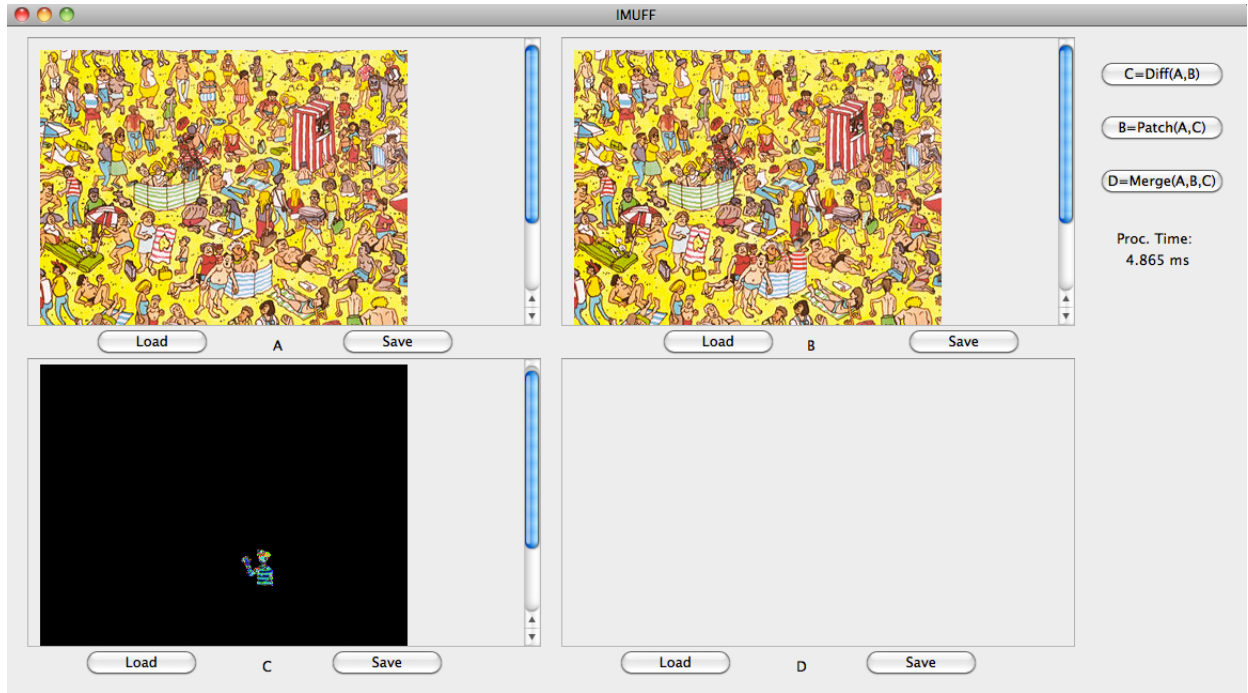


Figure 3.11: *IMUFF* toolkit used to perform *diff*, *patch* and *merge* operations over image artifacts. Available at: <https://github.com/gems-uff/gemuff>.

as baselines<sup>4</sup> as there are no specific VCS for image artifacts. In all these experiments, a collection of 50 different sizes of images was used, distributed according to Figure 3.12, totaling 211.6 MBytes of data.

In 76 iterations (number used to better demonstrate the repository outgrowth), one image is randomly chosen and a transformation is performed on it, generating a new revision. This experiment used a limited set of transformations, based on the typically scale used in a real graphics production pipeline. The transformations employed were: pixel area change, filter, rotation, and horizontal flip, the last three being global transformations.

Additionally, in order to demonstrate the importance of our image processing technique and the proposed data structure, the results are labeled as *V1* and *V2*. *V1* corresponds to the first version of *IMUFF* [32], without employing image analysis techniques. *V2* is the current *IMUFF* version [31], which uses an image analysis technique to detect global operations and employs our data structure for image storage.

Figure 3.13 presents a chart with the repository growth for the pixel change transformation. In this scenario, only a subset of  $32 \times 32$  pixels is randomly changed. From this

<sup>4</sup>Each VCS has a directory where the information about versioned artifacts was kept. Git stores this information in “.git/objects”, Mercurial does it in “.hg/store”, and SVN keeps this data in the “svn/db” folder at the server side.

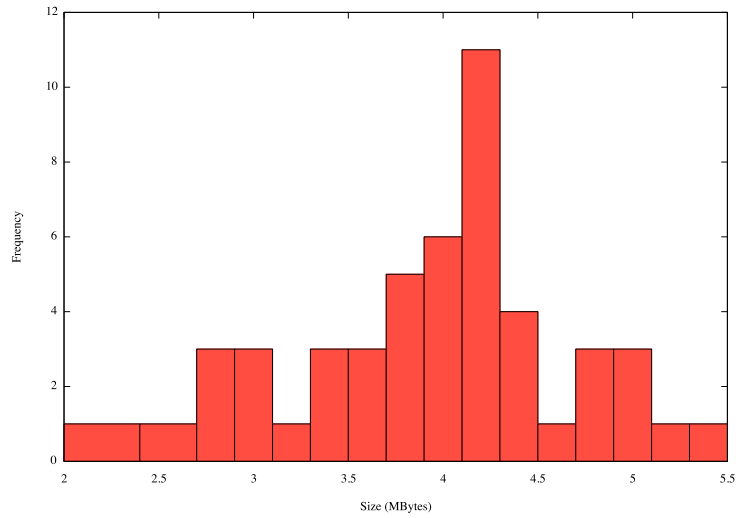


Figure 3.12: Image distribution in terms of size.

chart, we can clearly see that the *IMUFF* repository growth is the most reduced when compared to the other VCS between each revision, for both releases of our approach. While the other VCS solutions use 450–500MB by the end of the experiment, our approach ends up using only about 200MB. This shows that just storing what has been modified can considerably reduce the storage needs (as with *IMUFF*) when compared to saving the whole image for each revision (as done by the other VCS).

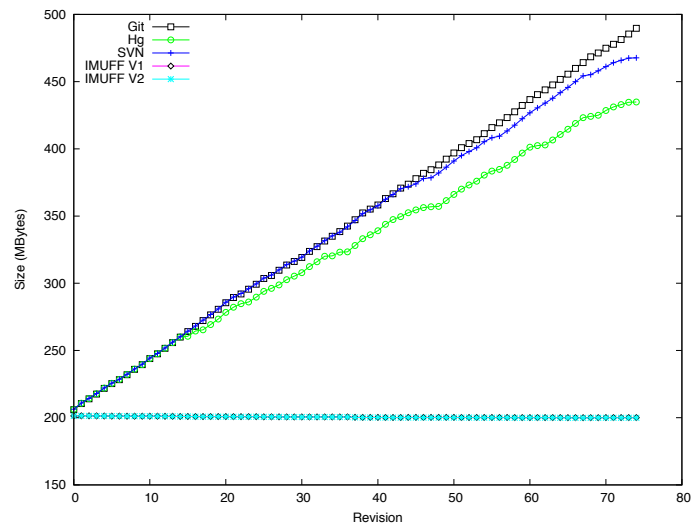


Figure 3.13: Image area change transformation (IMUFF *V1* and *V2* overlap).

In the sequence, in Figure 3.14(a) and Figure 3.14(b) both blur filter and a random rotation transformations are performed, respectively. According to these results, it is possible to see that Git, Mercurial, and Subversion are almost close in their repository

outgrowth<sup>5</sup>, while *V1* requires more data storage space to store *deltas*. This occurs because both filtering and rotation transformations change the whole image, with almost all the pixels in different colors in the *delta*, which degrades the data compression. On the other hand, an interesting fact occurs in the filter operation: it requires less space in each iteration for *V2*. This happens because blur is a mean filter, which tends to equalize the color using neighboring pixels. Thus, applying this filter sequentially to an image produces smaller deltas after compression. As *IMUFF* performs backward deltas in these examples, maintaining only the last revision as a whole, the space used by the last image revision gets smaller. For example, Table 3.2 shows the first three commits for a specific image used in this blur filter experiment. The first line represents the initial commit, leading to an empty delta (delta column) and a 6.5MB file (file size column), summing up 6.5MB. Before the second commit, a blur filter was applied to the image, reducing it to 4.3MB as well as a 1MB delta, leading to 5.3MB in total. In the third commit, another blur filter was performed, generating a new 4.1MB image and deltas adding to 1.1MB, with a total of 5.2MB data in the repository. It is worth mentioning that the cumulative size of the *deltas* is not reduced during the interactions. However, it rises more slowly than the decreasing rate for the image size, thus leading to a smaller repository size. Additionally, we should point out that, since it is a mean filter, blur decreases the image fidelity, producing more similar pixels in each interaction. Even with the loss of fidelity after successive uses of the blur filter, it is possible to build the original image by just applying each delta in the reverse order (i.e., from the last image to the first one).

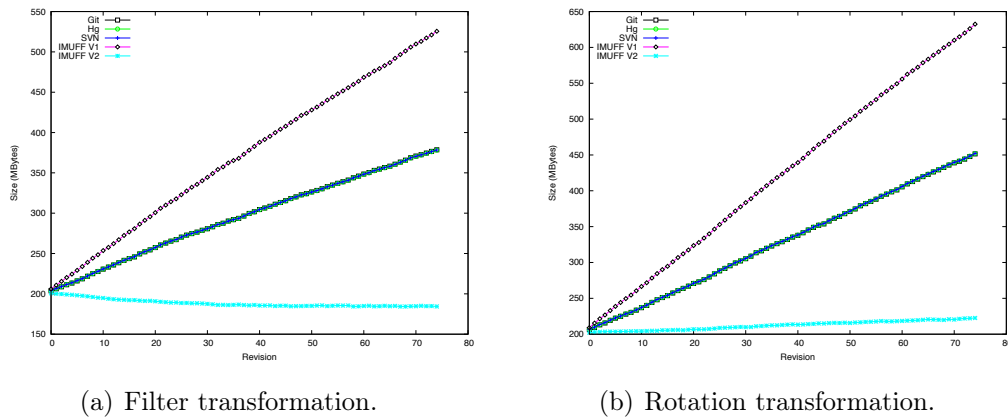


Figure 3.14: Applying global transformations: filter and rotation (Git, HG, and SVN overlap).

Additionally, the rotation operation benefitted from our image processing technique, leading to the smallest storage space requirement. As said before, the main benefit from

<sup>5</sup>Git, Mercurial and SVN grow in almost the same way: in the chart, their curves overlap.

Table 3.2: Evolution of a repository size for a specific file over three commits.

Commit #	File Size	Delta	Total
1	6.5 MB	-	6.5 MB
2	4.3 MB	1.0 MB	5.3 MB
3	4.1 MB	1.1 MB	5.2 MB

identifying rigid transformations is to recognize the semantics of a modification, such as translation, rotation, and/or reflection, instead of storing the plain *delta*. The knowledge that a rotation has been done to an image has an enormous impact on time and user confidence, since it removes the uncertainty of guessing a transformation by just looking at a change set.

Finally, Figure 3.15(a) and Figure 3.15(b) provide a chart with the horizontal flipping and mixed transformations, respectively. The horizontal flipping transformation is also a global image change that generates a *delta* consisting of all the pre-existing pixels. In this case, it is possible to see that *V2* has the most reduced storage space. The mixed chart consists of all the transformations distributed evenly (i.e., 19 transformations for pixel area change, filter, rotation, and horizontal flip). Using our first approach, the repository grows faster than all analyzed VCS due to the large number of global transformations, usually requiring more storage space. On the other hand, the newer version of our approach is the one that requires the least storage space of all analyzed VCS, thanks to its image analysis technique and data structure. Apart from reducing the storage space required by a repository, the bandwidth required to transfer data to distributed VCS, such as Git and Mercurial, would also be reduced as only the *delta* needs to be transferred.

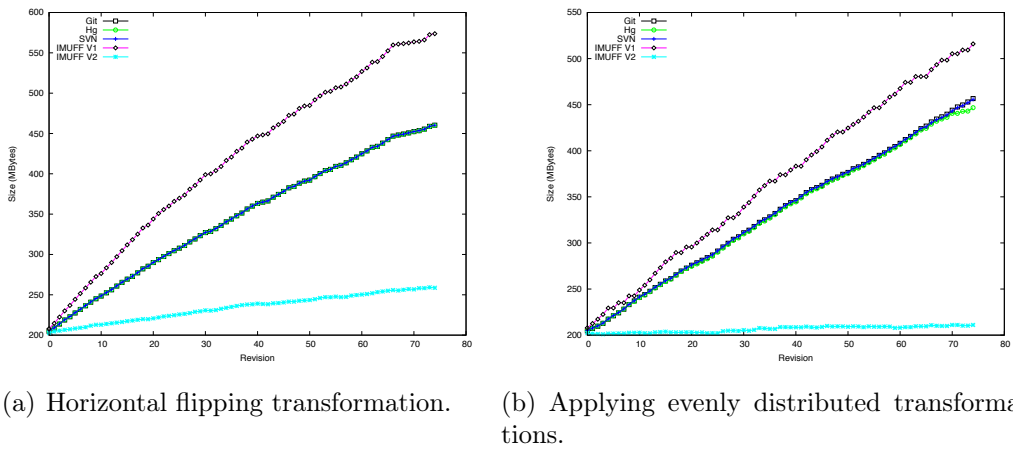


Figure 3.15: Applying horizontal flipping and evenly distributed transformations (Git, HG, and SVN are overlapped).

### 3.5.2 Performance measurement

This section presents the performance evaluation of our approach. For a fair comparison with a baseline, we implemented the *diff*, *patch*, and *merge* operations also in CPU. In the case of GPU, the time, measured in milliseconds, corresponds to the complete process, which is composed by the stages of opening and decompressing the image into the CPU memory, transferring the data from CPU to GPU memory, processing the data, transferring it back from GPU to CPU memory, compressing it, and saving it.

Table 3.3 shows the results from running these operations on the CPU and the GPU. According to the table, all operations that run in GPU are faster than the same ones done in the CPU. The column named *Speedup* shows how much faster the GPU version is in comparison with the CPU and is calculated as  $S = \frac{CPU_{time}}{GPU_{time}}$ . Also, in some cases such as the *merge* operation, an almost 55× boost is obtained. It is important to note that we are not using all the CPU cores in this evaluation, but only one. We believe it would be slightly faster if all CPU cores were used, but we should point out that traditional VCS do not run this process in parallel. Besides that, the overhead of managing a large set of threads that operates at the same structure at the CPU level may block an optimal parallelism at this architecture.

Another aspect one can observe in this table is that the *patch* operation is always faster than the *diff* operation with the CPU. After investigating this, we were able to see that the *deltas* used as inputs for *patch* consisted mainly of black colors, thus leading to a single representation of this color in the CPU cache memory. Finally, as expected, the *merge* operation is more resource-hungry than the other two operations as it consists of a *diff* followed by a *patch* operation. Moreover, it was faster than summing up *diff* and *patch*. Although the *merge* relies on both operations, context creation and data transfer from CPU to GPU and from GPU to CPU is done only once. Minimizing both operations while working with the GPU architecture greatly reduces the processing time [28].

Table 3.3: Speedup in running *diff*, *patch* and *merge* operations using CPU and GPU (time in milliseconds).

Image Size	Num pixels	Diff			Patch			Merge		
		CPU	GPU	Speedup	CPU	GPU	Speedup	CPU	GPU	Speedup
512×512	262,144	49.1	1.7	28.2	31.2	1.8	17.3	74.3	2.3	32.3
1,024×1,024	1,048,576	197.5	4.6	43.0	131.7	4.7	28.0	320.3	5.82	55.0
2,048×2,048	4,194,304	611.3	13.6	44.9	352.6	13.8	25.5	950.7	18.33	51.8
4,096×4,096	16,777,216	2,433.8	55.3	44.0	1,358.1	56.1	24.2	3,782.1	73.13	51.7

To better illustrate this speed boost, Figure 3.16 has a chart with the amount of time required to execute these operations with both the GPU (using *V2* of *IMUFF*) and CPU,

where time is shown in a  $\log_{10}$  scale. It can be seen that all the operations done by the CPU take almost two orders of magnitude more in comparison to the same operation run by the GPU.

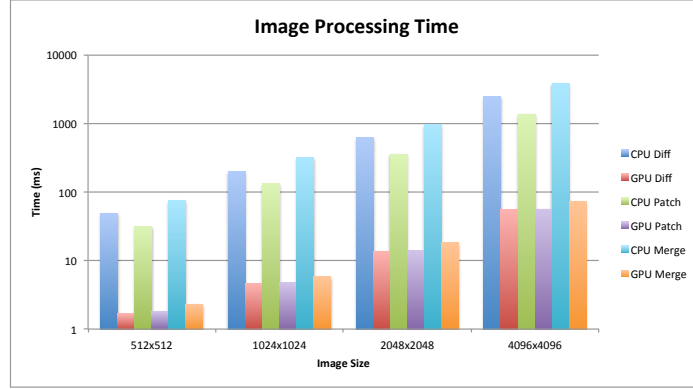


Figure 3.16: Comparison of *diff*, *patch* and *merge* operations as run by the CPU and GPU for four image sizes. Vertical axis is shown in  $\log_{10}$ .

Figure 3.17 shows the time required for a VCS check-in (i.e., commit command) for Git, Mercurial, and SVN compared to both versions of *IMUFF* in the evenly distributed transformations scenario (Figure 3.15) at each iteration. Again, note that *IMUFF* is not a VCS as-is, so the time considered here is just to run the *diff* operation, which involves applying the previous image algorithm and building or updating the data structure for each image. Also, it is important to note that Git only gets the whole image artifact for the revision requested, running a faster process, as the artifacts do not need to be reconstructed by running a *patch* operation. Also, the whole image is stored for each revision, thus not requiring any kind of processing (i.e., *diff* is not applied). On the other hand, both SVN and Mercurial use a binary *diff* for detecting blocks of different bytes by using *xdelta*<sup>6</sup>. For this reason, our approach is expected to use more time than all the other VCS, as the artifact is being processed. Furthermore, our last version requires more time to process than the first one, due to the image processing for rigid transformation extraction. However, should Git, Mercurial, and SVN use our approach, the *push* and *pull* commands would require less network traffic, as the *delta* has been shown to be smaller. This benefit requires extra processing effort during check-in.

<sup>6</sup>Website: <http://xdelta.org>



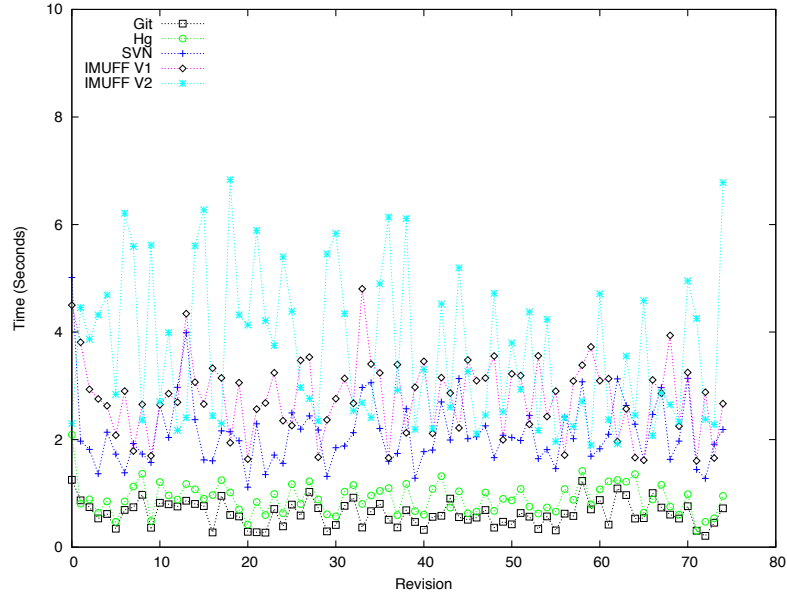


Figure 3.17: Time needed to make a check-in operation in the evenly distributed transformations case.

## 3.6 Threats to validity

People who deal with images normally use the set of image transformations that were evaluated in this thesis on a daily basis. However, to state how *IMUFF* performs in general, a real case study is desirable. On the other hand, according to the presented results, we believe that the only side effect from using *IMUFF* would be the extra processing time in the cases where an image transformation cannot be extracted.

Additionally, during performance evaluation, we did not consider all the available cores of a CPU. However, when comparing the number of cores found in a CPU, it could lead to a small fraction of the cores found in a GPU (which could be thousands). This way, our approach design is targeted to be executed by a GPU and its hundreds of cores, enabling the CPU to deal with other tasks such as version control for text-based artifacts. We believe that even when using all the cores of a CPU to run *IMUFF*, it would still be much slower than with a GPU.

## 3.7 Related work

Digital content management refers to the general process of authoring, editing, collecting, publishing, and distributing digital content, both in text and binary formats [63]. Among

the various components of digital content management, version control is one of the most important, as artifacts generally change over time.

Conventional version control mechanisms are mainly focused on text rather than binary data format, either on the low level (via line diff [59]) or the high level (via programming language syntax [62]). In the case of binary files, such as videos or images, it is common to store the files as a whole or crude binary difference in each revision, without any semantic information [59]. According to Babich's idea of team coordination [9], the maintenance of duplicate nearly identical images, as handled by the *merge* functionality of a VCS, is crucial to Computer-Supported Collaborative Work (CSCW) as well as a method of allowing conflict resolution. Dealing with images as binary artifacts does not provide a mechanism for conflict resolution. This issue may hamper the adoption of VCS to manage this kind of asset, although there is a significant demand for version control of these files. Some approaches and techniques have emerged for trying to help users with image artifacts. This section presents the related literature divided into two groups: techniques that extract differences from images and version control for images per se. According to a systematic mapping review in this topic, detailed in Appendix A, 17% of the papers present operation based approach (i.e., the steps produced by the user are stored) in order to deal with image artifacts. Besides that, all papers present some kind of tool in order to aid in the *diff*, *patch*, and *merge* process. Finally, none of the analyzed approach deals with VCS integration.

Perforce 4<sup>7</sup> allows side by side image comparison, without any support for *diff*, *patch*, or *merge*. The PixelNovel 5<sup>8</sup> plugin for Adobe Photoshop<sup>®</sup> editor can process difference on layer, allowing the simulation of a *merge* operation through pixel combination among these layers. Chimera [70] is a plugin aimed at storing high level information in relation to user's action over image artifacts, allowing a future analysis over these data. Unfortunately, the lack of a formal representation among these actions can be inefficient on VCS. As a consequence, such approaches force the user to use a specific editor that does the image versioning.

Grabler et al. [49] proposed a system to automatically manipulate photos using GIMP, an open source image editor, and create visually appealing tutorials. Unfortunately, this approach requires GIMP to record all the transformations applied in its command sequences, not allowing its inclusion in any external VCS. This restriction may impose a barrier among users that is not flexible to change its tool or even do not want to. In

---

<sup>7</sup>Perforce website: <http://www.perforce.com>

<sup>8</sup>PixelNovel website: <http://pixelnovel.com>

contrast, our approach can be plugged in any working pipeline, as it is not tied to a specific tool.

Hu et al. [58] introduced an approach to recover a semantically meaningful editing history from a source image and an edited one. Their technique supports the detection of global and local linear and non-linear color changes, the insertion and removal of objects, and cropping. Although it produced interesting results, their technique requires considerable processing time (about three minutes for a  $512 \times 512$  image resolution when compared to 0.094 seconds required by *IMUFF* for the same image resolution), and so is not feasible for version control.

Version control of images is growing as images are being massively used in many applications, such as in the Web, simulations, videogames, and medical analysis. Wong [102] introduced an approach to manage Web images that are developed concurrently, implementing a framework capable of performing *merge* operation on image files and providing a method for visual conflict resolution. In this approach, an intermediate step is necessary, as image pixel data are extracted and transformed into an XML file. After this extraction, pixels in the same position are checked for equality and actions performed according to results. Our approach does not use conditional statements, which accelerates the processing. Apart from that, on top of the *merge* operation, we also provide *diff* and *patch* to be used independently. Finally, merging a  $255 \times 259$  image takes 1 second in their approach, while in ours, it takes 0.083 seconds for the same image resolution.

Chen et. al. [26] introduced a tool built in GIMP capable of performing non-linear version control for image artifacts (i.e., operations can be undone in any order) by representing the operations among revisions as a Directed Acyclic Graphic (DAG). Another tool, LayerVault [71], can do version control for Adobe Photoshop files (.psd) using a collaborative environment. Unfortunately, they do not provide a benchmark for time performance or space used, being impossible to find out if their approach can deal with large amounts of data in a reasonable time. Moreover, both were conceived to work integrated with image editing tools, reducing the possibilities of supporting situations where the images are versioned together with other artifacts, making a change set of the whole project. Regarding space, we believe that just storing the history of operations performed in images through an image editor would require much less space than the *delta* proposed in this work. However, depending on the complexity of the operations, the reconstruction of the versions might be time-consuming.

## 3.8 Final considerations

We presented in this chapter an approach to transform conventional VCS into an image-aware VCS through the use of *IMUFF*. This approach makes parallel work possible for image artifacts, enabling merging, and indicating any physical conflicts that may exist. Moreover, it generates an interpretable *delta* by showing the user rigid transformations (i.e., translation, rotation, and flipping), which can help users understand how the images evolved. Finally, *IMUFF* can reduce the storage space needed, and consequently the network bandwidth required by some VCS commands at the cost of a slight increase of the processing time. We believe that organizations that use large amounts of images, such as the ones in the video games or computer graphics industry, would greatly benefit from the use of our approach.

However, the processing of image artifacts requires a significant effort, which can jeopardize productivity, as one may have to wait until the operations finish in order to continue working. Thanks to the use of a GPU, which is a highly parallel architecture widely available in almost any development environment, the time required to process image artifacts is minimized when compared to using a CPU. In our experiments, the usage of our GPU solution boosts VCS common operations over image by up to 55×, when compared to a CPU-bound architecture. Our proposal is designed and implemented using an architecture that allows its easy deployment in existing VCS.

# Chapter 4

## Video-aware version control

### 4.1 Introduction

Recently, the number of video artifacts produced is growing up fast. Due to the social media websites, it becomes common to make and share images, videos, and other kinds of multimedia artifact over the Internet. Websites such as YouTube <sup>1</sup> and Instagram <sup>2</sup> are completely turned to store videos and images, respectively, produced by anyone who wants to share experiences.

Absent from producing videos for fun, industries such as game development and film also produce a high amount of video artifacts. However, a key aspect here is the iteration they make over the file during production, by modifying it until the desired video is achieved. Thus, in order to keep the history of these changes for situations such as rolling back undesired effects, normally each version of the video is saved under different names and versions.

The most popular VCS found today can be used for controlling the evolution of video artifacts. However, they do not provide specific operations to deal with this type of artifacts. In other words, operations such as *diff*, *patch*, and *merge* will have no effect when applied over video artifacts. Besides that, any file that cannot be dealt by VCS is saved as binary artifact due to their compression strategy, i.e., the whole file is saved for each revision unless a binary *diff* algorithm is employed. Even worse is the fact that the user cannot know what kind of modification was done between versions. The lack of specific operations to deal with video artifacts transfers the responsibility of controlling changes over these artifacts to the user, where the VCS acts just like a backup repository

---

<sup>1</sup>Youtube Website: <http://www.youtube.com>

<sup>2</sup>Instagram Website: <http://www.instagram.com>

for each version.

Normally, depending on the duration of the video as well as how it is saved, it can occupy a large amount of space to be stored. Due to that, some VCS hosting services, such as Github, do not allow a single file to be over 2GB <sup>3</sup>.

In this chapter we present an approach called *Video Multimedia UFF* (VIMUFF), which applies our developed concepts to video artifact as first-class elements in the context of version control. Our approach customizes the VCS base operations (i.e., *diff*, *patch*, and *merge*) to deal with this video artifact. As well as done in *IMUFF*, parallel maintenance is allowed through the *merge* support. Additionally, our contextual *delta* has the main purposes of allowing video change identification for the end user as well as reducing the size of the data to be maintained. In order to minimize the processing time, *VIMUFF* employs a parallel architecture based on a graphics processor unit (GPU) to implement *diff*, *patch*, and *merge* operations on videos.

Initially, Section 4.2 introduces the concept of digital video. In Section 4.3, we discuss the proposed approach for the solution while in Section 4.4 this approach is evaluated. In Section 4.5 some threats to validity are presented. Finally, Section 4.6 presents the related work and Section 4.7 presents the final considerations of this chapter.

## 4.2 Digital video

In its basic form, a video artifact can be considered a collection of still images that are presented in an ordered sequence for a certain amount of time. Due to the expansion of cheaper storage media, new compression techniques, high transmission rate, among others, the availability of them in different areas grows fast. Videoconference, education, training, monitoring systems, and entertainment are just few examples of situations that use video artifacts.

The following characteristics can be observed over video artifacts:

- **Resolution:** as well as image artifacts, the amount of space occupied by a video on the screen is determined by its resolution, which is defined by its width and height, in pixels. Figure 4.1 shows resolutions commonly used during video production.
- **Frame rate:** represents the number of images shown in a second. Highest frames

---

<sup>3</sup>Distributing Large Binary: <https://help.github.com/articles/distributing-large-binaries/>

per second (FPS) produces smoother movements to video. Theatrical movies are shot both with film and video typically at 24 FPS [20].

- **Codec:** due to the high number of information contained in a video artifact, a codec strategy is used to compress such information. Additionally the same codec used during a video compression needs to be used during decompression for showing the video.
- **Container:** such as image formats, video artifacts also have standard formats, normally corresponding to its extension. Some of these formats includes *AVI*, *MOV*, and *FLV*. It is important to state that not all video that has the same extension (container) uses the same codec.

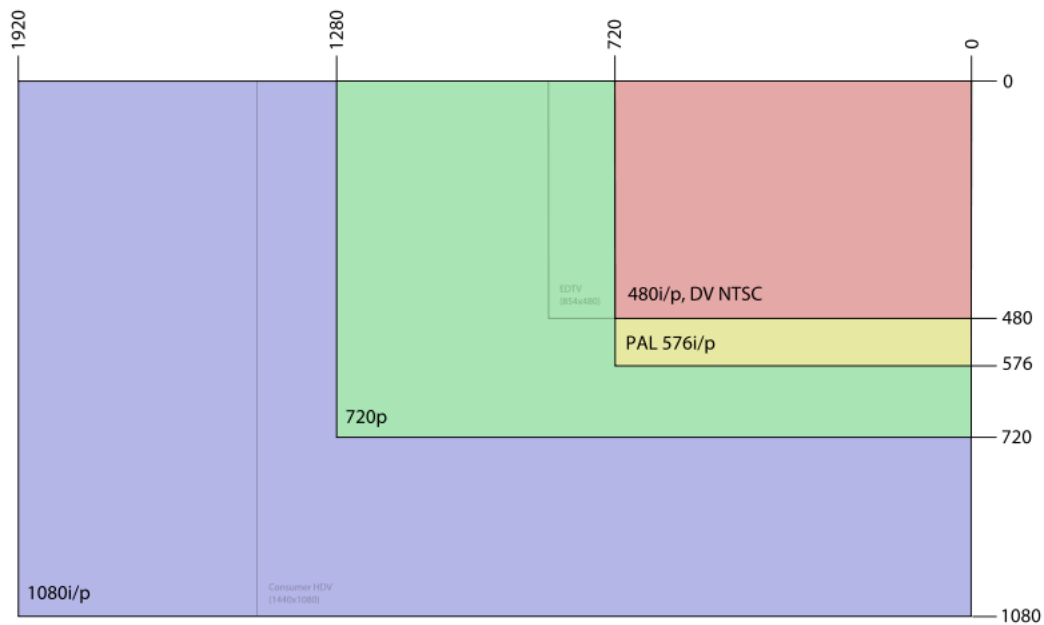


Figure 4.1: Common resolutions for video artifacts (width  $\times$  height) [68].

One of the biggest problems in relation to video artifacts is due to its high amount of data. In order to better illustrate, ten minutes of an uncompressed colored video (18,000 frames) leads to an amount of 3 Gb of data, assuming a  $256 \times 256$  video resolution [6]. Following the same idea, ten minutes of the same video in Full HD ( $1920 \times 1080$  resolution) occupies about 149 Gb. Efficient transmission and storage can be done by using some standard compression techniques [72, 77]. In this case, processing this kind of artifact requires decoding it before extracting and analyzing the frames.

### 4.3 VIMUFF *diff*, *patch*, and *merge* operations

In this section, we present an approach for processing video artifacts based on specialized *diff*, *patch*, and *merge* operations. Besides that, we also present the data structure used to store the *delta* as well as how it is internally organized.

#### 4.3.1 *Diff* operation

Given two specific videos, extracting their difference consists of locating where new frames were added or removed. To do so, we propose the process described in Figure 4.2.

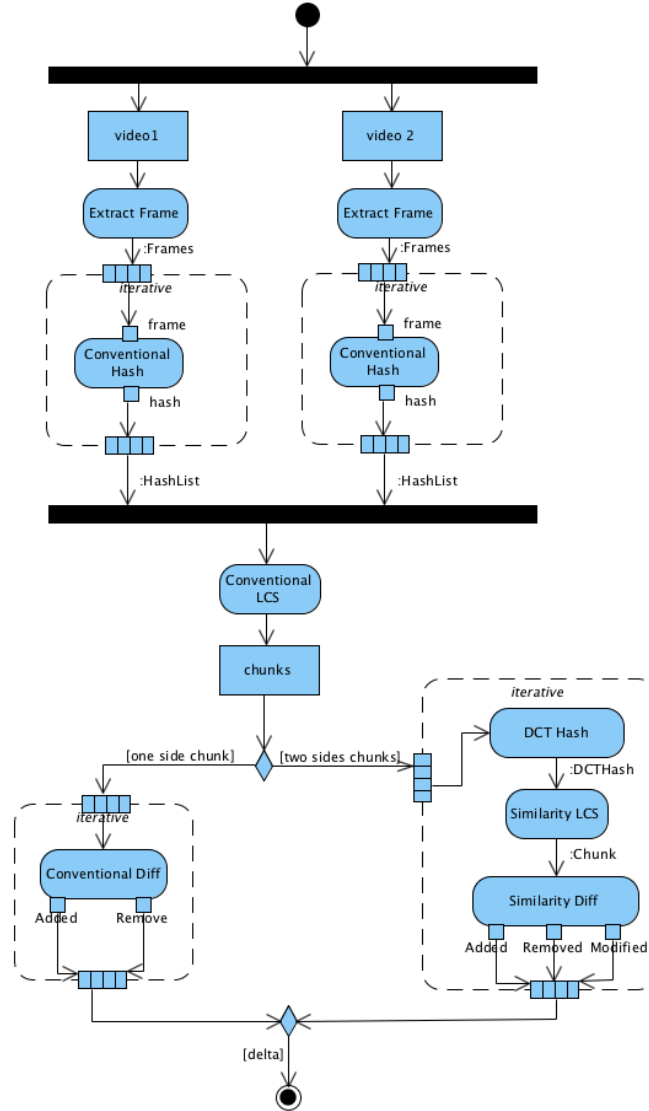


Figure 4.2: Activity diagram for processing a *diff* in VIMUFF.

This process starts by extracting the video's frames as set of images (*Extract Frame*). However, depending on the *codec* used for producing the video, extracting these frames



can vary considerably on its strategy. For example, some *codecs* store a frame as the difference from its predecessor and not as a full image, while others may do this process just for specific frames (called key frames). In order to support a wide range of videos, the FFMPEG <sup>4</sup> library was adopted for extracting video's frames from different *codecs*. These frames are extracted in a chronological order as they appear in the video.

Unfortunately, depending on the video's length, the number of frames extracted can be relatively high, consuming considerable amounts of memory. Considering this fact, each frame is indexed by a hash, which is based on its content. Afterwards, only the hash is processed to detect changes in the video. This hash is built by *Generate Hash* process, producing an MD5 hash [92]. The aforementioned process is illustrated in Figure 4.3, where the letters on top of each frame represents their identification.

Based on the fact that the same frame can appear more than once in the video, thus having the same hash, it is stored just once and a hash list containing the sequence of the frames' hash used during the video playback is created, as shown in the bottom of Figure 4.3. This example shows that frame A is used both at beginning and end of the video, according to its sequence list, but in fact this frame is stored just once in the memory. Both extracting and generating the hash of the frames to produce the hash list are made in parallel for each one of the videos.

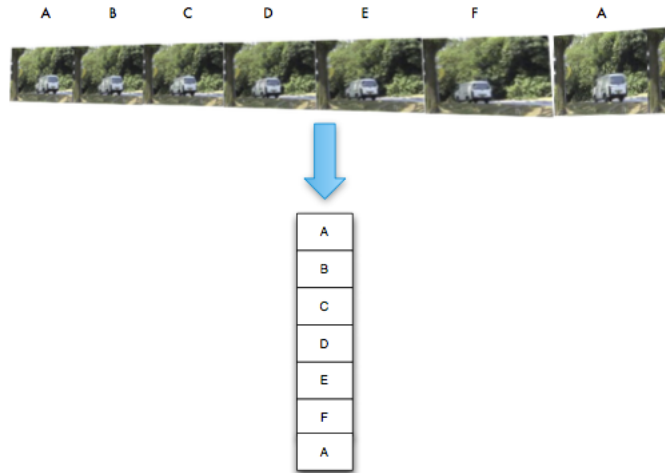


Figure 4.3: Identification and list sequence generation in *VIMUFF*.

At this point of the process, our approach produced two lists containing the sequence of hashes of the frames for each video. Locating the difference between these two sequences is now a matter of processing them in order to check which frames are common to both videos. This is made by using the *Longest Common Subsequence* (LCS) [57] algo-

<sup>4</sup>FFMPEG website: <http://ffmpeg.org/index.html>

rithm. The LCS algorithm is responsible for extracting the common largest subsequence of elements between two lists and is performed by the *Conventional LCS* step.

Finally, after computing the LCS we can process the difference (identify added and removed frames) by contrasting the LCS with each one of the videos (*Conventional Diff*). This *diff* calculation is responsible to classify the sequence of frames in video 1 or video 2 that do not appear in the LCS. These sequences can be classified as added if they appear on video 2 or as removed if they appear on video 1. Figure 4.4 shows two sequences of frames and their difference. According to this example, four frames are common to both videos while three frames were removed and four frames were added. Due to the fact that *VIMUFF* uses a directional delta, the *delta* between these two videos would be built by indications for removing B, Y, and D frames from video 1 and adding U, K, W, and G frames in the correct position of the same video. In this case just frames U, K, W, and G would be fully saved in the *delta*, as well as some additional data indicating their position for insertion and instructions indicating which frames to remove to compose video 2 during a *patch* operation.

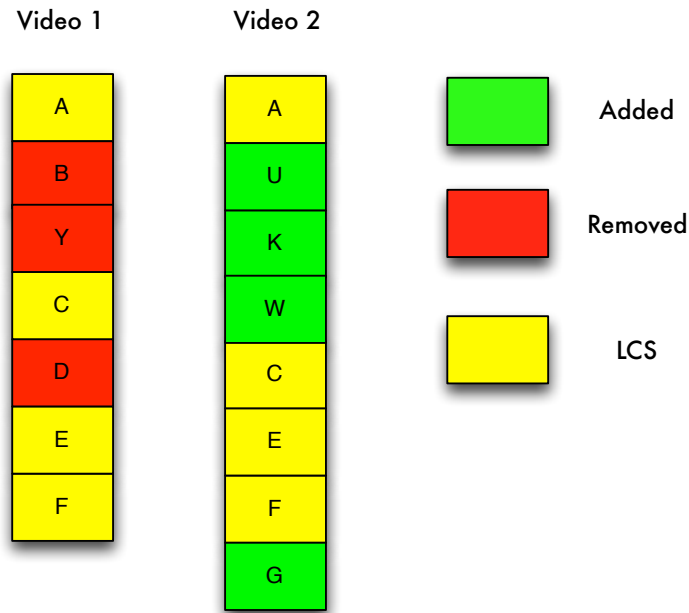


Figure 4.4: The difference between two video's sequence. Nodes colored in yellow are common to both videos, while red nodes represents frames not in *video 2* (removed) and green nodes represents frames just in *video 2* (added).

One problem that may occur is related to small changes to a video, such as adding a subtitle. In this situation, all frames that do not have the subtitle will be marked as removed, followed by the same frame, but with the subtitle, marked as added. In this

case, storing all these added frames will increase the *delta* size significantly.

For this situation, *VIMUFF* is also capable of detecting modifications among frames besides addition and removal, allowing to better understand changes performed on the video as well as decreasing the *delta* size as just the difference between the two frames is stored. According to Figure 4.2, after producing the LCS from the two hash lists, the *DCT Hash* process is invoked when a pair of chunks (frames that are between the LCS) between the two videos is found. This process is responsible to produce a hash list for each frame in the chunk, for different chunks. However, this new hash produced has the property of being comparable to other hash to check how close they are. In the sequence, the *Similarity LCS* is triggered in order to calculate an LCS considering similarity among frames in chunks. Finally, *Similarity Diff* process is invoked to process frames identified as modified, added, or removed to produce the *delta*.

Figure 4.5 shows a screenshot of *VIMUFF* tool during its processing. In the tool, the first video appears at the top while the second in the middle. Additionally, the *delta* between them is shown at the bottom. Figure 4.5(a) shows frame added (using green border) in the delta. On the other hand, Figure 4.5(b) shows frames removed (using red border), while Figure 4.5(c) shows modified frames (using a yellow border).

Measuring how different is a given frame from another requires the usage of a perceptual image hash function. These functions produce values based on the visual appearance of the image, allowing such hashes to be compared in order to compute how similar two images are.

There are techniques that use different functions operator to produce hash identification, such as the Discrete Cosine transform (DCT), Marr-Hildreth, radial variance, and block mean value [103]. In this thesis, the DCT is used for detecting similarity among images, similar to the one used on pHash [103] library.

The DCT, such as any Fourier-related transform, is responsible to express a function or signal in terms of a sum of sinusoids, each one having different frequencies and amplitudes. The DCT just use cosine functions to express a signal.

#### **Definition 4.3.1. DCT**

Let  $x[m]$ ,  $m = 0, \dots, N - 1$  denote an  $N$ -point real signal sequence. Strang [98] defines DCT as

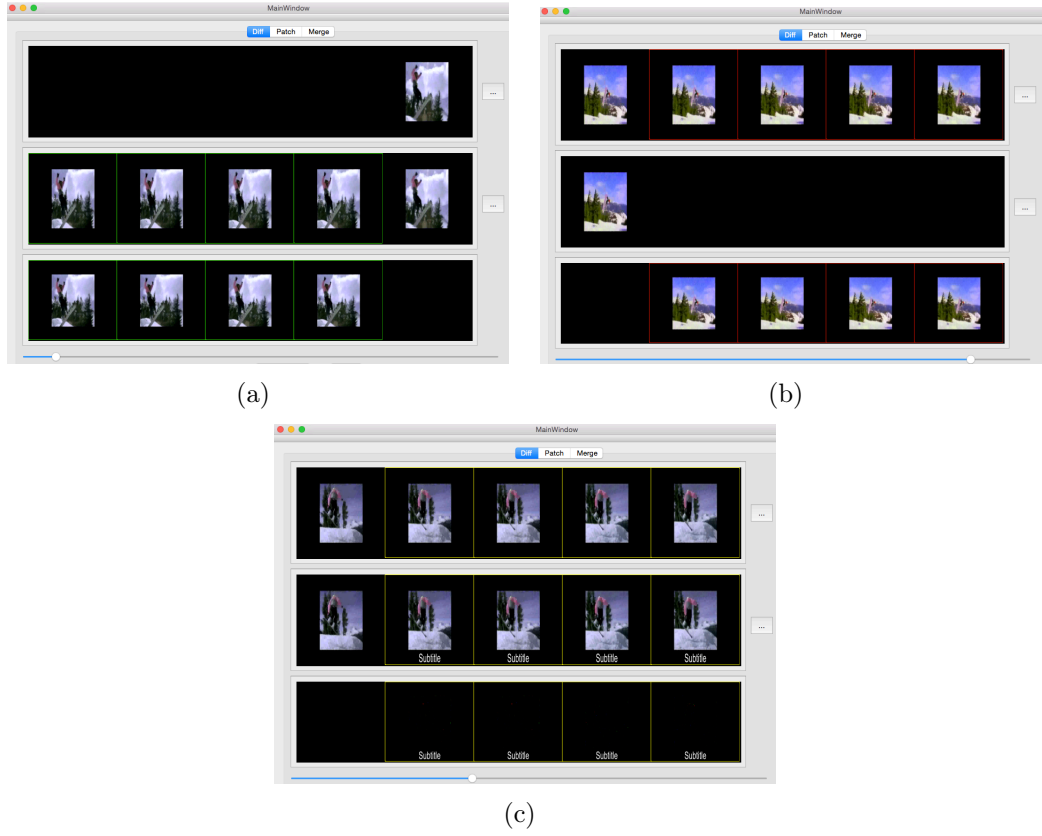


Figure 4.5: *Diff* detection by *VIMUFF*. In (a) frames were added while in (b) frames were removed, and finally in (c) frames were changed.

$$X[n] = \sqrt{\frac{2}{N}} \cdot \sum_{m=0}^{N-1} x[m] \cdot \cos\left(\frac{(2m+1) \cdot n\pi}{2N}\right), (n = 0, \dots, N-1). \quad (4.1)$$

Equation 4.1 can be expressed as

$$X[n] = \sum_{m=0}^{N-1} c[n, m] \cdot x[m], (n = 0, \dots, N-1). \quad (4.2)$$

where  $c[n, m]$  denotes the row number  $n$  and column number  $m$  of the DCT matrix, defined in Equation 4.3.

$$c[n, m] = \sqrt{\frac{2}{N}} \cdot \cos\left(\frac{(2m+1) \cdot n\pi}{2N}\right), (m, n = 0, \dots, N-1). \quad (4.3)$$

As DCT is a separable linear function, it can be applied to any two-dimensional image first along one dimension and then to the other dimension. In this case, the DCT of a two-dimensional image  $I$  can be computed as ( $M$  denotes the DCT matrix in Equation 4.3)

$$DCT(I) = M \cdot I \cdot M' \quad (4.4)$$

According to Lin [74], low-frequency DCT coefficients of an image are mostly stable under image manipulations, as most of the signal information tends to be concentrated in a few low-frequency components. After applying the DCT formula over the image, the element close to the top-left (index position (0,0)) represents the low frequency components therefore being perceptually most significant. Coefficients located in higher vertical and horizontal indexes represent higher frequency components. Figure 4.6 shows two examples where it is possible to observe the variation of the DCT according to how images are composed. Figure 4.6(b) is composed by abrupt changes so its DCT information is spread far from the beginning of the image (index position (0,0)). On the other hand, Figure 4.6(a) presents smooth changes, having its frequency information concentrated on the beginning of the image.

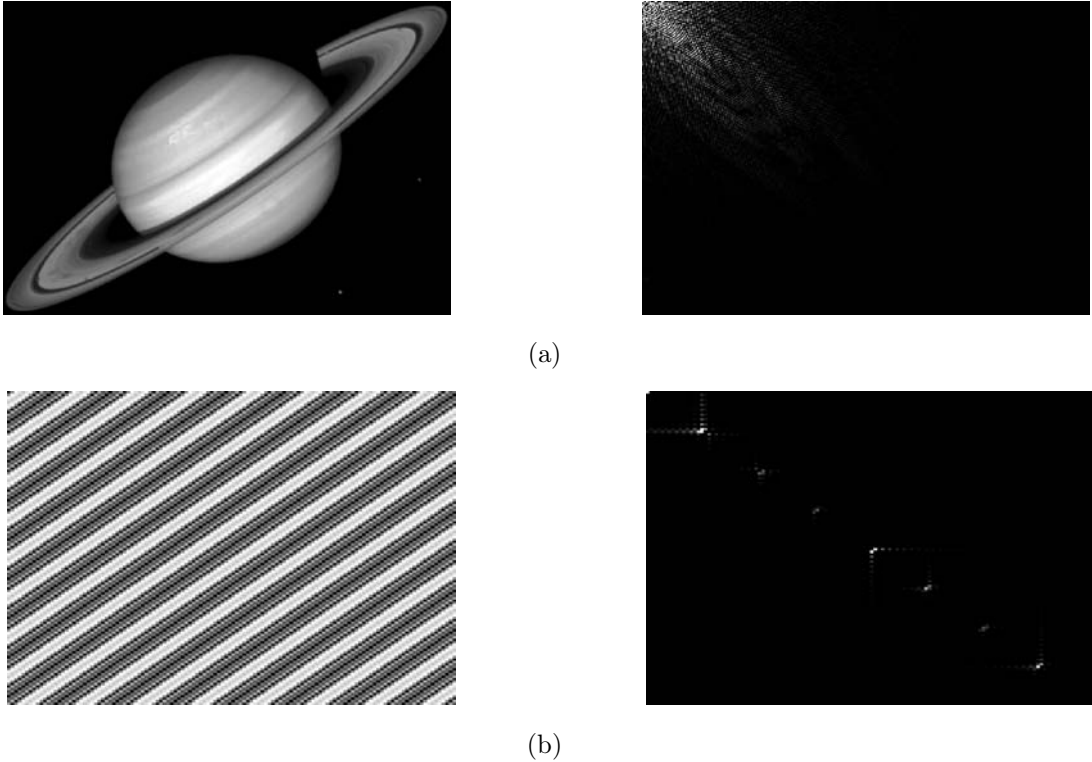


Figure 4.6: DCT processing over two different images. The image in (a) has a small slope between tones changes while in (b) tones change abruptly. Images taken from [66].

Besides that, instead of computing the DCT from a RGB colored image (RGB stands for red, green, and blue), it must be first converted to gray scale, where the essential semantic information resides in. Additionally, the image is reduced to a  $32 \times 32$  resolution. In order to avoid high frequencies, a  $7 \times 7$  mean filter is applied over the image before

its reduction. After this process, the DCT formula is applied over each row and column of this image and just  $8 \times 8$  (as the four quadrants are the same) samples are used in order to compute the hash. Algorithm 4 presents the execution of this step as well as the extraction of a 64-bit hash.

**Data:** image  
**Result:** 64 bits hash

```

1 ConvertToGrayScale(image, grayScaleImage);
2 Apply7x7MeanFilter(grayScaleImage, grayScaleImageFiltered);
3 ResizeImage(grayScaleImageFiltered, grayScaleResized, height:32, width:32);
4 DCT(grayScaleResized, dctImage);
5 CropImage(dctImage, dctCropped, h:8, w:8);
6 float mean = MeanValue(dctCropped);
7 ulong64 one = 0x00000000000000001;
8 ulong64 hash = 0x0000000000000000;
9 for  $i \leftarrow 0$  to 64 do
10   if GetFloatValue(dctCropped,  $i \bmod 8$ ,  $i \bmod 8$ ) then
11     hash |= one ;
12   end
13   one = one « 1;
14 end

```

**Algorithm 4:** Algorithm for processing DCT hash based.

According to Lin [74], feature code can be extracted from the relationship between two DCT coefficients in the same position of different images. In this case, similarity could be calculated by comparing each of these coefficients for different images and calculating how distant they are. This is performed through the Hamming Distance [52], which is basically a measurement for the difference of two given strings [103].

#### Definition 4.3.2. Hamming Distance

Let  $A$  denote an alphabet of finite length.  $x = (x_1, \dots, x_n)$  denotes an even-length string, whereas  $x \in A$ . The same holds true for  $y = (y_1, \dots, y_n)$ . Then the hamming distance  $\Delta$  between  $x$  and  $y$  is defined as

$$\Delta(x, y) := \sum_{i=1}^n |x_i - y_i| \quad (4.5)$$

#### Definition 4.3.3. Normalized Hamming Distance

In order to facilitate comparison, the hamming distance can also be normalized with respect

to the length  $n$  of the strings. Swaminathan [99] defines the normalized hamming distance  $\Delta_n$  as

$$\Delta(x, y)_n := \frac{1}{n} \Delta(x, y) \quad (4.6)$$

The hamming distance can be calculated for strings representing either a number system, such as binary, or alphabets. As an example, Table 4.1 shows three different kind of strings. In the first line, a binary system is used so the difference from 0 to 1 is one and occurs once. On the second line, a decimal system is used and two digits are different between the first and second string by one value, summing two. Finally, an alphabet string is used, having a hamming distance of four (going from letter *a* to *e* requires passing over four letters, i.e., **b**, **c**, **d**, **e**).

Table 4.1: Examples of calculating the Hamming distance. In the first row, a binary string is used while in the second a decimal string is used. Finally, the third string shows an alphabet [103].

String 1	String 2	Hamming Distance
00101	10101	1
12345	13344	2
well	wall	4

In this case, calculating the hamming distance between two binary coded numbers can be done by applying the XOR operation. Denoting  $a$  and  $b$  as two binary coded numbers of equal length, the hamming distance is defined to be the number of ones in  $a \oplus b$ .

Due to the amount of processing that needs to be done, a DCT hash is just calculated from chunks between two LCS frames. As an example, supposing that frames B and Y were subtitled, producing frames K and W, respectively, in Figure 4.4, the *similarity diff* would find out those modifications when processing the sub-list between A and C. As a result, just frame U would be considered added while frames K and W would be considered as modification, as shown in Figure 4.7.

Both processes *DCT Hash* and *Similarity Diff* presented in Figure 4.2 can be further decomposed as presented in Figure 4.8. The first step after calculating the LCS is performed by *Create Subsequence* step, responsible to create a hash sub-list for each frame in each chunk. This process produces a total of  $n$  sub-lists for each video ( $SL^1 \dots SL^n$ ), being  $n$  the total chunks of the video. According to the figure, it is executed in parallel for each video.

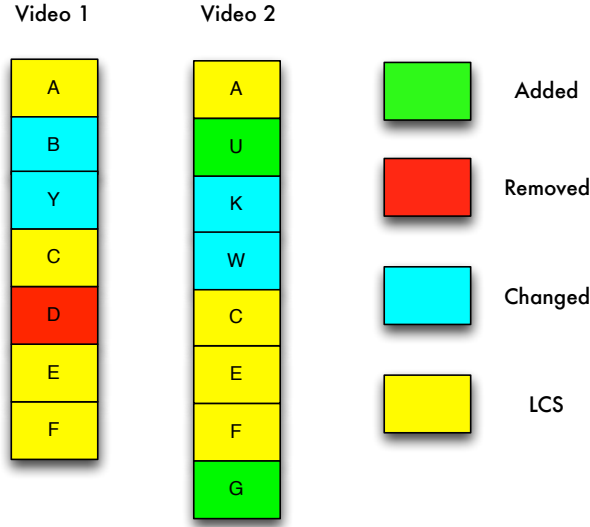


Figure 4.7: The difference between two video's sequence. Nodes colored in yellow are common to both videos, while red nodes represent frames not in *video 2* (removed) and green nodes represent frames just in *video 2* (added). Cyan nodes represent frames that suffered modifications.

After producing the sub-list, three possible situations can occur during processing:

1.  $SL_{video1}^i.Size() > 0$  and  $SL_{video2}^i.Size() = 0$   $\therefore$  this represents frames removed and no further processing is necessary.
2.  $SL_{video1}^i.Size() = 0$  and  $SL_{video2}^i.Size() > 0$   $\therefore$  this represents frames added and no further processing is necessary.
3.  $SL_{video1}^i.Size() > 0$  and  $SL_{video2}^i.Size() > 0$   $\therefore$  this represents potentially modified frames and requires analyzing frames on  $SL_{video1}^i$  and  $SL_{video2}^i$  for similarity.

The first and second cases are straightforward and require just marking those frames as added or removed (*Process Added Frames* and *Process Removed Frames*, respectively). However, the third case involves further processing for detecting frames similarity. When the third case happens, a similarity hash must be produced for each frame in the sub-list. However, as stated before, a series of manipulation is first needed before generating this hash (*Convert Frames to Gray*, *Apply Mean Filter*, *Resize Frame*, *Apply DCT*, and *Crop Image*). After all these manipulations are done, the hash can be generated using the resulting image. At this stage, each sub-list pair go through *Similarity LCS* processing for producing a similarity LCS. During processing, two frames are considered similar if their hash distance is equal or below a threshold, which can be configured by the user during processing.



Finally, after producing the similarity LCS, the *Process Modified Frames* are triggered, in order to locate the position of each LCS in the sub-lists. It is important to notice that it is not possible to assume the frames are located on the same index in both sub-lists as they may have different lengths. One example of this situation can be found in Figure 4.7, where the sub-list between LCS A and C will have two frames for video 1 (B and Y) and three frames for video 2 (U, K, and W). Frames in sub-lists that are not similar to other are processed as added or removed.

In Figure 4.8, some steps are done in CPU and GPU. Actions colored in yellow are executed on GPU as they involve transformations that are performed over frames, while actions colored in blue are performed on the CPU. Algorithm 5 shows the whole execution of the processing.

```

Data: Video 1, Video 2
Result: delta between Video 1 and Video 2
1 sequenceListV1 = nil;
2 sequenceListV2 = nil;
3 LCS = nil;
4 while Video 1 has more frames do
5   RegisterImage(v1-current-frame, hash);
6   AddFrame(sequenceListV1, hash);
7 end
8 while Video 2 has more frames do
9   RegisterImage(v2-current-frame, hash);
10  AddFrame(sequenceListV2, hash);
11 end
12 ProcessLCS(sequenceListV1, sequenceListV2, LCS);
13 currentV1Idx = 0;
14 currentV2Idx = 0;
15 for lcs-node  $\leftarrow$  first(LCS) to last(LCS) do
16   LocateLCSNode(lcs-node, sequenceListV1, idx1);
17   LocateLCSNode(lcs-node, sequenceListV2, idx2);
18   if idx1 - currentV1Idx > 0 and idx2 - currentV2Idx > 0 then
19     v1SubSeq = FramesFrom(sequenceListV1, currentV1Idx, idx1);
20     v2SubSeq = FramesFrom(sequenceListV2, currentV2Idx, idx2);
21     SimilarityLCS(v1SubSeq, v2SubSeq, simLCS, delta);
22   end
23   else if idx1 - currentV1Idx > 0 then
24     FramesRemoved(delta, sequenceListV1, currentV1Idx, idx1);
25   else if idx2 - currentV2Idx > 0 then
26     FramesAdded(delta, sequenceListV2, currentV2Idx, idx2);
27 end

```

**Algorithm 5:** *Diff* algorithm considering added, removed, and modified frames.

It is important to state that the difference between two modified frames is processed by using *IMUFF diff*, as presented in Section 3.4.1, i.e., using a binary **XOR** ( $\oplus$ ) over each channel for all the pixels of both images.

#### 4.3.1.1 Data structure

In order to store the *delta* during a *diff* operation and further use it to reconstruct a new revision of a video artifact during a *patch* operation, a new data structure is needed. When considering video artifact, it is necessary to specify the right position in the sequence where each modification happened. Considering two sequences  $S_1$  and  $S_2$  representing two videos, the  $diff(S_1, S_2)$  operation gives an index where modifications happened in the sequence  $S_1$  as well as the kind of modification (*OP\_ADD*, *OP\_REMOVE*, or *OP\_MODIFY*).

When a frame is present in  $S_1$  and not found in  $S_2$ , just the index of the frame and a flag indicating that it was removed is necessary to be stored. However, when a frame is present in  $S_2$  and not in  $S_1$ , it is considered as an addition and the frame content must be stored in the *delta* to allow the reconstruction of the respective version. Finally, when the similarity LCS find that two frames are similar, *IMUFF* is triggered in order to calculate the difference between frames, saving just the index and the difference itself (performed by *SimilarityLCS* in Algorithm 5). As the difference between two similar frames tends to have most of the pixels in black, the *delta* size is reduced.

In order to clarify these steps, Figure 4.9 presents a complete example. In Figure 4.9(a), a modification is made in video 1 to produce video 2. In this example, the bold number in the top left corner presents the index of the frame in the sequence. According to the figure, the LCS between both videos are A and C. When analyzing the frames between A and C, we can observe that frame U was added while frames K and W were identified as modification over frames B and Y in video 1, respectively. Also, after frame C we can notice that frames D, E, and F were removed and frame G was added. This whole situation produces a *delta* according to Figure 4.9(b), representing the difference between those two videos.

### 4.3.2 Patch operation

VIMUFF uses directional delta for composing revisions of video artifacts. Applying a *patch* in order to retrieve a version is done according to Equation 4.7.

$$Version_j = Patch(Version_i, \Delta_{i \rightarrow j}) \quad (4.7)$$

As Figure 4.9 illustrates, storing video 1 as a whole and just the *delta* between video 1 and video 2, constructing the latter requires the following steps: add frame U at index 1; modify frames B and Y at indexes 1 and 2, respectively; remove frames at index 4, 5, and 6, respectively; and finally add frame G at index 7.

However, during video 2 reconstruction, it is possible that specified indices in the *delta* become no longer valid. For instance, when adding frame G at index 7 (the last operation specified by the delta), the video produced so far will have only five frames. It is due the fact that adding and removing frames changes the absolute index specified in the delta. In order to avoid this problem, an *offset* is always added to the index specified by the delta. This *offset* is incremented or decremented by one when a frame is added or removed, respectively. Algorithm 6 illustrates all the steps performed during a *patch* operation. As observed, when a modified frame is found, *IMUFF* is triggered to process the resulting patched frame.

<p><b>Data:</b> Video<sub>i</sub>, Δ<sub>i→j</sub>  <b>Result:</b> Video<sub>j</sub></p> <pre> 1 offset ← 0; 2 while Δ<sub>i→j</sub> has more actions do 3   index ← GetIndex(currentAction); 4   frameData ← GetData(currentAction); 5   if Operation(currentAction) = OP_ADD then 6     InsertFrame(frameData, index + offset, Video<sub>i</sub>); 7     offset ← offset + 1; 8   end 9   else if Operation(currentAction) = OP_REMOVE then 10    RemoveFrame(index + offset, Video<sub>i</sub>); 11    offset ← offset - 1; 12  else 13    IMUFFPatch(FrameAt(Video<sub>i</sub>, index + offset), frameData); 14  end 15 end </pre>
--

**Algorithm 6:** Patch algorithm considering frames added, removed, and changed.

### 4.3.3 Merge operation

The *merge* operation is responsible for conciliating two modifications made in parallel over a common video artifact. In this case, it is necessary to combine both modifications in order to produce the final video.

Considering a video 1 artifact and two modifications made in parallel over it, producing variants video 2 and video 3, the *merge* operation produces a video 4 containing modifications performed on both video 2 and video 3. This process involves the execution of a set of steps in order to merge these videos. Besides that, instead of using the *diff* specified in Section 4.3.1, which considers just two artifacts for extracting the difference, here the three way *diff* (called *diff3*) is used. *Diff3* extracts the difference considering a common ancestor, thus allowing a more precise way to recognize conflicts. Specifically for VIMUFF, a conflict is caused when two modifications over the same set of frames between two consecutive LCS are made.

Calculating the *diff3* among video 2 and video 3 based on the ancestor video 1 involves first calculating the LCS between video 1 and video 2 (called  $LCS_{1 \rightarrow 2}$ ) and between video 1 and video 3 (called  $LCS_{1 \rightarrow 3}$ ). In the sequence, a new LCS is performed between  $LCS_{1 \rightarrow 2}$  and  $LCS_{1 \rightarrow 3}$ , producing  $LCS_{2 \rightarrow 3}$ , which contains common frames among video 1, video 2, and video 3. After that, the *diff3* continues verifying what was changed between  $LCS_{2 \rightarrow 3}$  frames in order to process addition, removal, and modifications.

Figure 4.10 presents an example of *diff3* processing. Frames colored in yellow on video 2 and video 3 represent the LCS between the respective video and video 1 ( $LCS_{1 \rightarrow 2}$  and  $LCS_{1 \rightarrow 3}$ , respectively) while yellow frames on video 4 represent common frames to all videos ( $LCS_{2 \rightarrow 3}$ ). It is important to notice that space between frames is desired for aligning the versions. According to the figure, it is possible to observe that the final video (video 4) will have frames X and K between LCS frames A and B as just video 2 changed (added frames) in this subsequence between those frames. The same occurs between frames B and D as well as between frames D and F as one frame is removed from video 2 and a modification is made in video 3, respectively. However, when checking between frames F and I, it is possible to see that both videos (2 and 3) changed frames G and H, thus causing a conflict. As done by any VCS, a conflict situation requires the user intervention in order to select which version to use in order to produce the merged video.



Figure 4.8: Expansion activity diagram for both *DCT Hash* and *Similarity Diff* processes. Yellow activities are done in GPU.

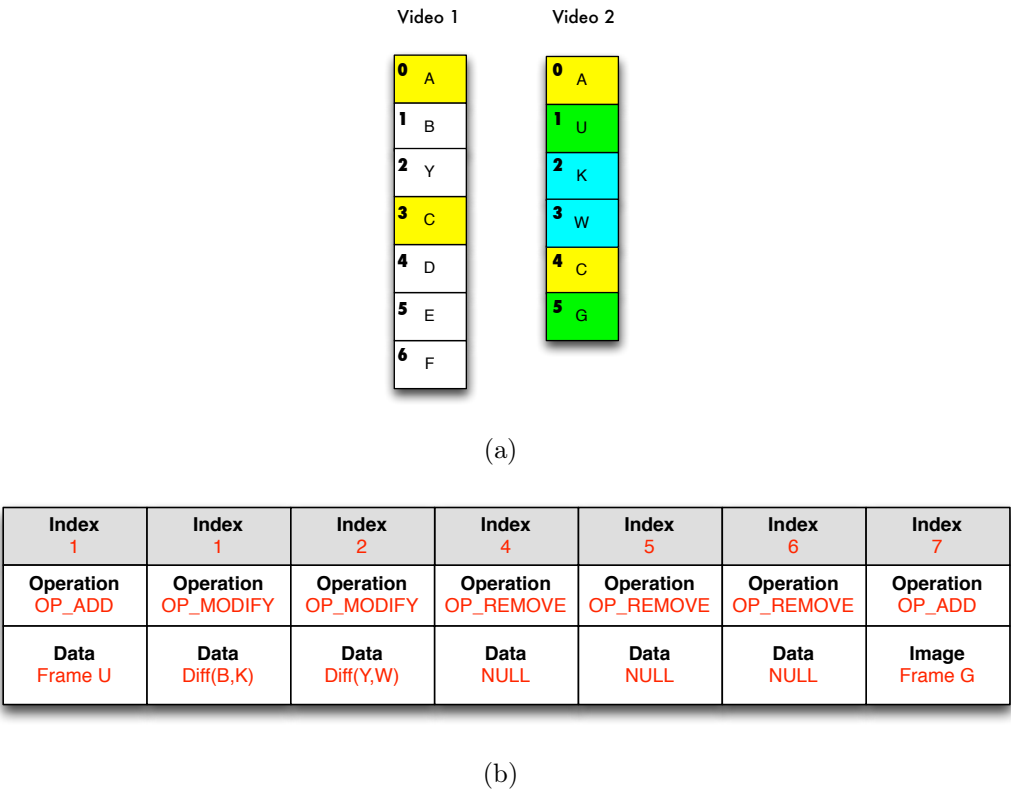


Figure 4.9: Example of how *delta* is organized. In (a) video 1 suffered a modification, where yellow, cyan, and green represent LCS, modified, and added frames, respectively. In (b) the data structure that represents these operations.

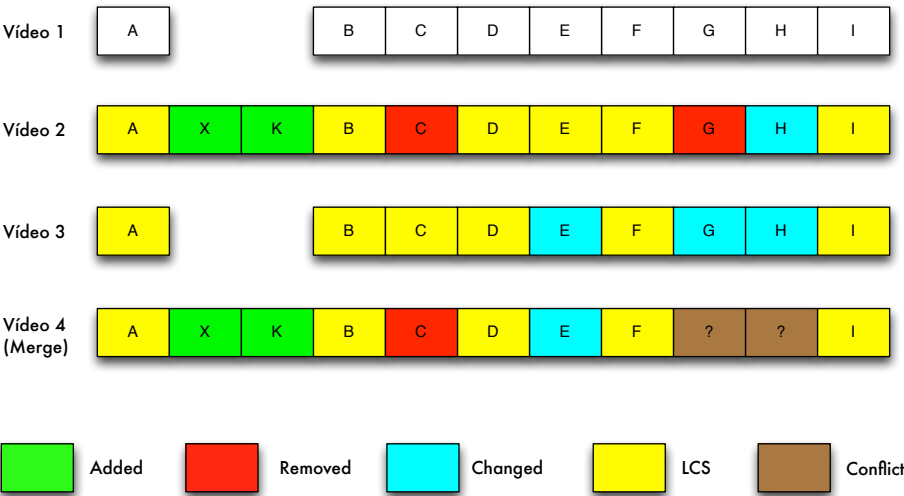


Figure 4.10: Merge of two videos *B* and *D* descending from *A* and producing final video *E*.

## 4.4 Evaluation

This section presents an evaluation of our approach in terms of space consumption and performance. For all experiments, an Intel Core i7 4.4GHz PC with 16GB RAM and an nVidia GeForce GTX580 graphics card was used. This GPU has 16 Stream Multi-processors with 32 Stream Processors each, giving up to 512 cores. For the evaluation, we developed a GUI (Graphical User Interface) to activate the process and inspect the generated results. Table 4.2 shows the properties of the video used for evaluation. It is important to state that all the videos are in  $624 \times 352$  resolution and 24 FPS in uncompressed format. Additionally, frames are considered similar when they have a minimum of 70% of similarity (maximum distance of 30%).

Table 4.2: Properties of the videos used during the experiments.

Video	Duration	Size (MB)	Number of Frames	MD5 Hash (ms)
A	4 min. 48 sec.	2,280.00	6,923	16,674
B	4 min. 0 sec.	1,897.00	5,060	15,690
C	4 min. 0 sec.	1,877.00	5,699	16,341
D	1 min. 0 sec.	474.50	1,440	2,288
E	1 min. 0 sec.	474.50	1,440	2,194

The evaluation description is divided into two sections. In Section 4.4.1, the evaluation focuses on data storage space. In Section 4.4.2, the experiments observe processing time in both GPU and CPU.

### 4.4.1 Storage space

Normally, one of the reasons to use *delta* is to reduce the storage space requirement. Instead of saving two versions of an artifact, just the difference between them is stored.

In order to analyze how *VIMUFF* impacts storage space requirements for video artifacts, Table 4.3 shows the number of frames removed, added, and changed as well as the *delta* size. It is important to state that the removed frames almost do not impact our *delta* size as just a flag and an index is necessary to be stored. On the other hand, frames added impact in the size of the *delta* as they must be fully stored. Finally, the number of modifications has a small impact over the *delta* size as just the difference between frames must be stored.

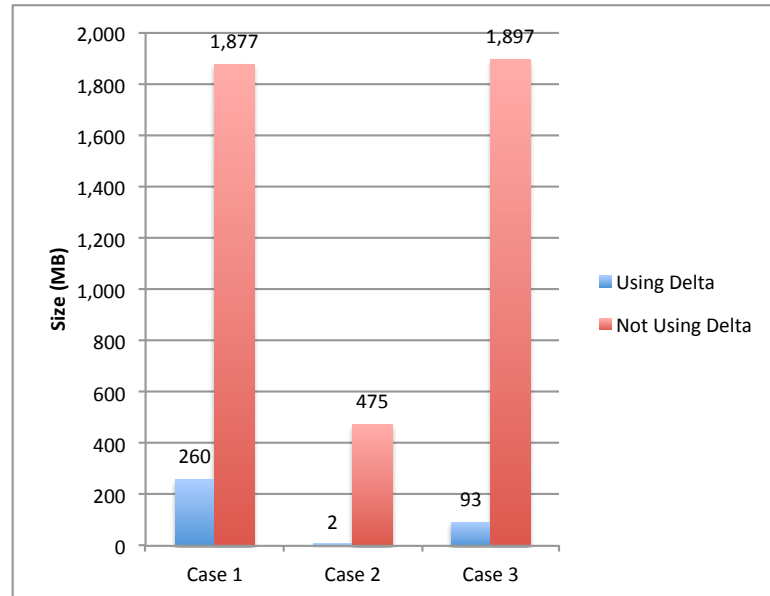
Our *delta* requires less space than storing the video as a whole, as can be seen in Figure 4.11 for each case presented in Table 4.3. According to this figure, our *delta* reduces the

Table 4.3: Frames removed, added and changed as well as the *delta* size for each case.

Case	Operation	Frames			Delta (MB)
		Removed	Added	Changed	
1	Diff(A,C)	1,329	2,553	280	260
2	Diff(D,E)	14	14	255	1.7
3	Diff(C,B)	1,379	1,440	500	93

space consumption in 86.15%, 99.57%, and 95.09% for the Diff(A,C), Diff(D,E), and Diff(C,B), respectively. The fluctuation on this percentage is due to the number of frames added and modified, as removed frames just require storing index and flag information. When the number of added frames grows, the size of the *delta* tends to increase as well as each frame needs to be fully stored. This is represented by Diff(A,C) and Diff(C,B) cases, when compared to the Diff(D,E) case. On the other hand, modified frames tend to not increase the size of the *delta* as just the difference between the frames are stored, as can be seen by Diff(D,E) case. Also, notice that even when the Diff(C,B) has almost twice of modified frames in relation to the other two cases, its *delta* size still remains below the Diff(A,C) case.

This analysis is made by only considering one revision. When considering multiple revisions of the video, this number tends to increase faster.

Figure 4.11: Comparing using *delta* and not using it for storing video artifacts.



### 4.4.2 Processing time

As presented in Figure 4.2, our approach is divided into a set of steps, performed in an ordered sequence. Figure 4.12 shows the amount of time spent for each step during processing all cases presented on Table 4.3 (using GPU). As it is possible to observe, the *Conventional Hash* generation step is the most expensive one, followed by *Conventional Diff* processing. Also, it is important to notice that all steps performed for processing similarity diff (*DCT Hash*, *Similarity LCS*, and *Similarity Diff*) require substantially less time when compared to the other steps.

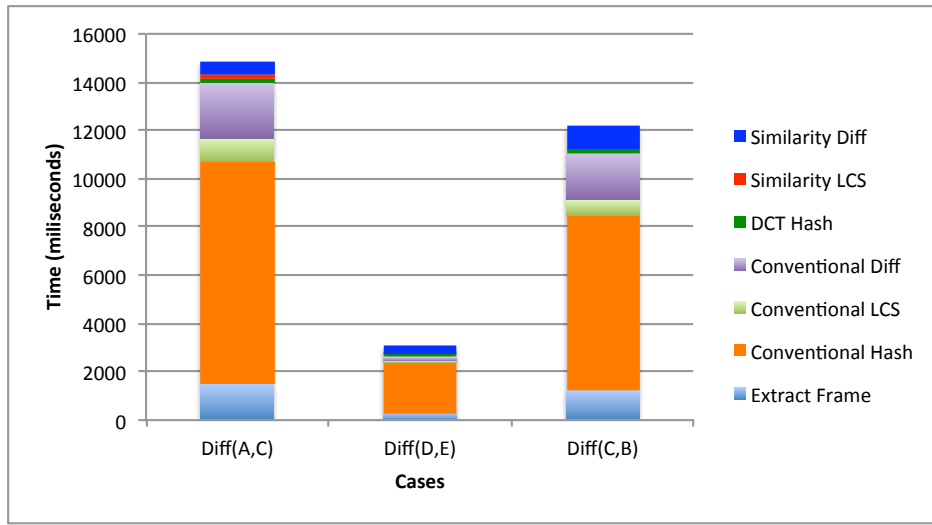


Figure 4.12: Time spent for each step during a *diff* processing using GPU.

According to Figure 4.8, just a number of steps are performed on GPU when similarity diff is being processed. To show the difference of both GPU and CPU proposals, we compared the execution of these steps as presented in Figure 4.13. In this figure, all the steps used for the *DCT Hash* generation are compared between GPU and CPU. Moreover, the *Similarity Diff* processing corresponds to cases where images are processed through *IMUFF* for extracting the *diff* between two frames.

When looking at these cases, it is possible to observe that processing DCT Hash is slightly increased on GPU. The reason for this is related to memory latency, as frames are processed sequentially one by one in GPU. To avoid this latency, a batch of frames should be submitted concurrently to the GPU. Also, as these times represent just cases where a possibly modification is found between chunks, it is expected to spend more time on situations where it happened most, represented by *Diff(C,B)* (a total of 500 frames modified). Besides that, it is possible to observe that the time taken for processing similarity on GPU is almost three times faster than same processing on CPU.

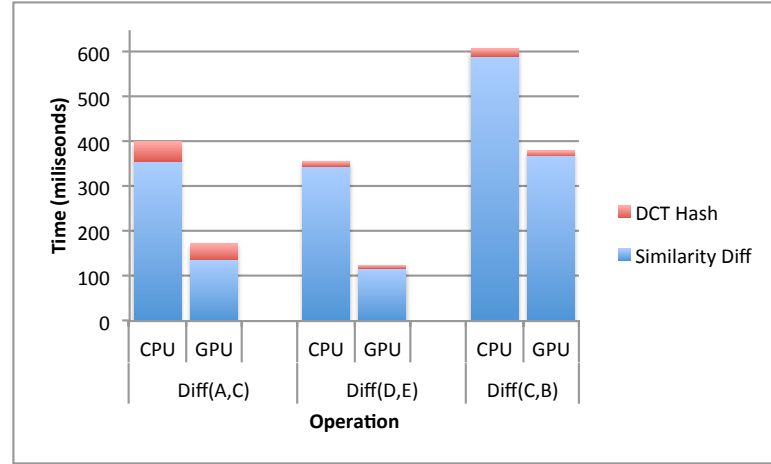


Figure 4.13: Execution time for processing the *DCT Hash* and *Similarity Diff* between two versions of a video using GPU and CPU.

In Figure 4.14 is possible to observe the results of the *patch*:  $Patch(K, \Delta_{K \rightarrow Y})$ . This figures shows that the same performance obtained when processing the *diff* operation on GPU is achieved, which means that the *patch* operation on GPU reduces to the half of the time taken by processing at the CPU. Moreover, the *patch* operation does not need to calculate LCS, as reconstructing a version is just a matter of removing, adding or processing modified frames. In this case, the most expensive operation is to process a modified frame, as it needs to go over pixel by pixel in order to apply the *delta*.

In the first case ( $Patch(A, \Delta_{A \rightarrow C})$ ), it takes 198ms on GPU and 440ms on CPU. When looking at the second case ( $Patch(D, \Delta_{D \rightarrow E})$ ), the CPU version requires 337 ms to process the *patch*, while in GPU it requires 127ms. Finally, the third case ( $Patch(C, \Delta_{C \rightarrow B})$ ) requires 273ms to process in GPU while in CPU it requires 593ms.

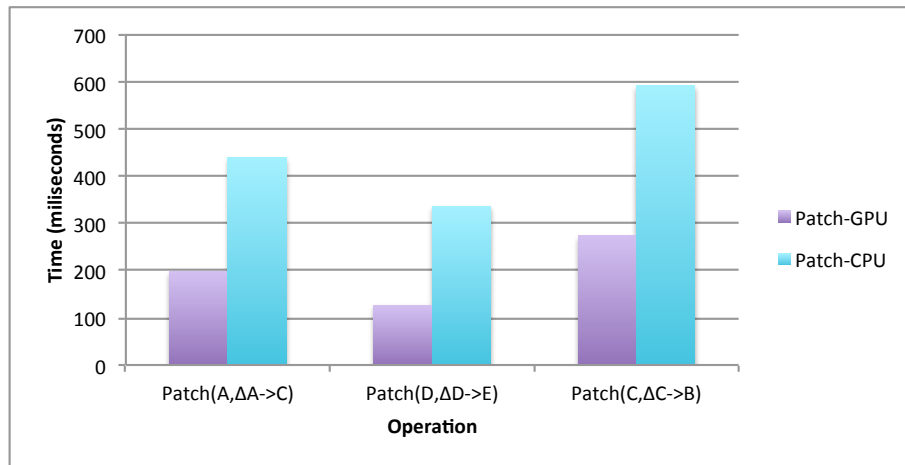


Figure 4.14: Execution time running a *patch* operation.

Finally, for the *merge* operation, we have used video A as base and videos B and

C as parallel modifications. As said before, this operation requires the *diff3* to work properly, which involves calculating LCS three times. All steps as well as the time taken for each one can be seen in Figure 4.15 for the GPU version. It is possible to observe that the most expensive processing step is regarding to *Conventional Hash* followed by *Conventional Diff3*.

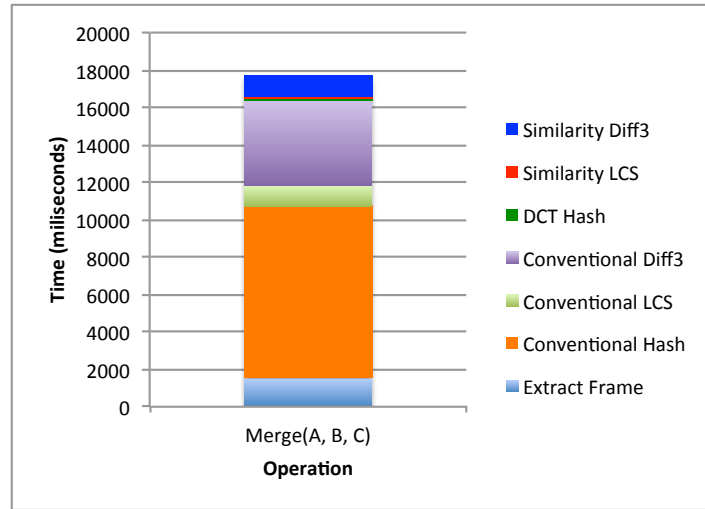


Figure 4.15: Time spent for each step during a *merge* processing using GPU.

In order to analyze just the steps where GPU is employed, Figure 4.16 presents the time for both the *DCT Hash* and *Similarity Diff3* processing. As observed, processing the *DCT Hash* and *Similarity Diff3* in GPU is 2.55 and 2.18 times faster, respectively. Different from the *diff* where the *DCT Hash* processing has slightly increased when compared to CPU, here the memory latency was hidden by GPU processing as the number of chunks tends to increase during a *merge* operation (reduced number of LCS frames).

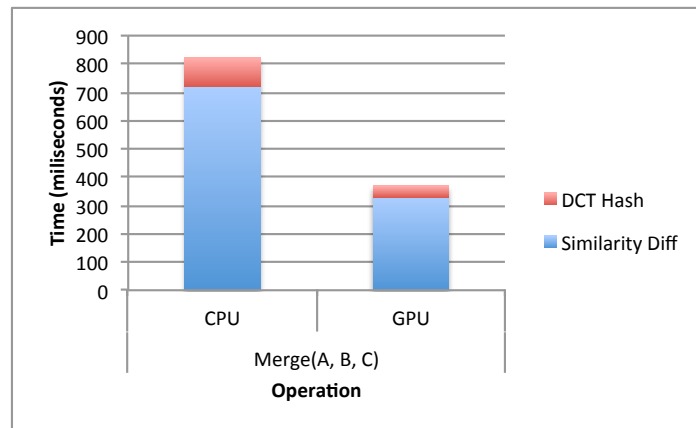


Figure 4.16: Execution time for processing the *DCT Hash* and *Similarity Diff3* during a *merge* operation using GPU and CPU.

## 4.5 Threats to validity

Although *VIMUFF* can work with compressed video formats, all the videos used in the experiments were uncompressed. Compressed video size is highly reduced when compared with non-compressed videos at a price of lowering the quality of the video.

Additionally, the number of videos used during evaluation was kept to a minimum, specifically for the *merge* evaluation. More videos should be used in order to observe how *VIMUFF* behaves with them.

Finally, we did not present any case where all frames are added from one version to another. This case should be interesting to evaluate how the *delta* increases.

## 4.6 Related work

In our systematic mapping review we could not be able to find any research directly related to versioning of video artifacts, i.e., proposing specialized operations for *diff*, *patch*, and *merge*. The usual approach for versioning video artifacts is storing the versions as a whole during each modification, without any awareness of the kind of modification that was made.

Considering high level editing tools, some of them allow a workaround for merging two video artifacts. For instance, Adobe Premiere<sup>®</sup> can combine two videos taking into consideration the weight that each pixel has over the other, just like image alpha blending. Using this technique, it is possible to produce the merged video.

Boar<sup>5</sup> facilitates the storing of large binary files. According to them, Boar is considered a “*version control for large binary files*” and allows the manipulation of just a small subsets of binary data at a time, so maintaining control of which part is being manipulated and by who. However, Boar does not provide any kind of awareness about modifications made over these files, and does not allow merging parallel modifications. On the other hand, *VIMUFF* can work with video artifact just like standard VCS can work with textual artifacts by allowing merging and providing awareness about modifications performed over these kind of artifacts.

Some plugins help Git to better deal with large binary artifacts. *Git-annex* [47] allows managing large files without the need of committing the file into the repository. In this

---

<sup>5</sup>Boar website: <http://www.boarvcs.org/>

case, the content of the file is kept by *git-annex* in a distributed key/value repository. The file that actually gets checked into Git is just a symbolic link to the real file. Git-Media [24] is another extension to Git with the same goal. For this, a new command is appended to Git that allows synchronizing only those specific large files.

## 4.7 Final considerations

In this chapter we presented an approach named *VIMUFF* to allow conventional VCS to process video artifacts. It generates interpretable *delta*, where users can see which modifications were performed. Besides that, it allows the merging of these artifacts as well as indicating any physical conflicts that may exist, relying on the user to choose which version to maintain.

Additionally, we had shown that storing the whole file for each revision can consume a great amount of disk space when compared to storing just the difference between each revision. In this thesis, we had shown that, depending on the size of the video and the modifications performed on them, almost 95% of data storage is saved. This represents a considerable amount of saved disk space, making more viable a complete version control of video.

Due to the amount of data that needs to be processed, it must be fast in order to not jeopardize the user productivity. In fact, we show that performing *diff*, *patch*, and *merge* operations is faster in GPU than the same processing in CPU.

# Chapter 5

## Exploratory data analysis of software repositories

### 5.1 Introduction

When working on software projects, developers often need to answer numerous questions, such as: “which other methods do I need to edit if I make this change?”; “who was the developer that last edited this method?”; “who do I need to coordinate my changes with?”; “who is the expert in a specific file?” and so on [44].

Besides that, software repository analysis can also be used to identify expertise. Expertise identification in a software project is an important issue for task allocation, personnel hire, onboarding, and development help, among other activities. It has been observed that when stuck in a task, developers often use their implicit knowledge of work dependencies to identify a developer who can help [56], or rely on their social network to find others who might know enough about the artifact in question [80]. In fact, managers often use informal processes to facilitate their team members to come talk to them (e.g., a manager keeping a candy bowl in his office), so that they are aware of who is having what kinds of problems and can direct allocate developers to help each other.

Some of these questions can be answered by doing a repository exploration. However, such kind of exploration is not a trivial task, especially when there is an extensive amount of data that is accrued over the project lifecycle and when this data is stored across different repositories.

In order to overcome such problem, some approaches rely on filtering the data. For example, EEL [82] scopes the analysis to 1,000 project elements when identifying exper-

tise in a team, thereby restricting the application to smaller chunks of data. Besides that, performing a coarse-grain analysis, such as the file level, is a common approach as suggested by Cataldo et al. [23]. One problem and a possible inaccuracy of performing analysis at this level is that a developer may be recommended as an expert of the whole file, even if she only intensively worked on a small portion of that file. Additionally, inaccurate answers can be presented for posed questions when working at coarse-grain or scoped data. Analysis at the finer-grain (method or lines-of-code), however, leads to scale issues. Finally, some other approaches overlook evolution, such as Expertise Recommender (ER) [79], which considers the entire project history to make recommendation. Approaches based on overlooking the evolution do not consider that artifacts evolve over time as well as developers that may change their roles. Further, temporal analysis can show how expertise of a development team changes and whether there are artifacts that lack experts at a given moment in time.

In this chapter propose Dominoes [64, 30, 29] for allowing a new realm of explorations over software repositories. In order to be efficient, allowing large-scale repository analysis at interactive rates, Dominoes adopts the parallel architecture of GPU to process the underlying data much faster than what can be possible with CPU processing [3]. Dominoes<sup>1</sup> is a framework that can be used by other tools for processing relationships [34] or by our provided GUI. Dominoes GUI is a visual tool for allowing users to play with their own repository for performing different kinds of exploratory analysis.

In Section 5.2 we start by introducing our approach, Dominoes, designed to enable interactive and exploratory analysis. In Section 5.3 two examples of Dominoes applicability are presented: one for relationship identification and another for expertise extraction over an artifact. Additionally we show how coarse and fine grain can impact over our analysis, including a temporal analysis. In Section 5.4 we applied the previous two examples over a real project, Apache Derby<sup>2</sup>, considering dependency, granularity, expertise and its evolution. Also we evaluate the performance of Dominoes in the same section. In Section 5.5 we evaluate Dominoes GUI regarding its usability. In Section 5.6 threats to validity are presented. Section 5.7 presents the related work among data repository exploration and expertise extraction. Finally, in Section 5.8 we present our final considerations.

---

<sup>1</sup>Available at: <https://github.com/gems-uff/dominoes>.

<sup>2</sup>Derby Repository: <https://github.com/apache/derby>

## 5.2 Dominoes

Dominoes aims at enabling users to explore the relationships among their project elements. Our approach organizes data from a software repository into multiple matrices that are treated as domino tiles, such as [developer|commit], [commit|method], [class|method], amongst many other combinations. Just as in the Dominoes game, where joining two congruent squares edge to edge can form a rectangle, our matrices can be combined to create additional (derived) matrices.

Dominoes allows exploring relationships across different levels of granularity. Besides that, it allows extracting temporal developer expertise by considering modifications over parts of an artifact. Therefore, the granularity aspect is a central architectural element (e.g. [package|class], [file|class], and [class|method]). Connecting any other domino tile with these composition tiles or their transposed tiles allows navigation from coarse-grained to fine-grained analysis or vice versa. However, fine-grained analysis can lead to extremely large data sets to be analyzed. In order to solve the scalability problem, Dominoes implements the exploratory analysis of software project entities as linear algebra operations over matrices, which can be parallelized in GPU architecture [69]. This allowed boosts in performance of about three orders of magnitude. Therefore, Dominoes opens a new realm of exploratory software analysis, as endless combinations of domino tiles can be experimented and generated in an exploratory fashion.

We denominate a matrix as  $M$  and its transpose by using a superscript ( $M^T$ ). Individual elements in a matrix are denoted as  $M[i, j]$ . The operator “ $\times$ ” represents matrix multiplication. It is important to note that when multiplying two matrices the number of columns in the first operand must be equal to the number of rows in the second operand. In our case, the column and rows of the operand over which we are multiplying also need to be congruent (same project element), similar to the Dominoes game. In other words, we can multiply [developer|commit]  $\times$  [commit|method], but not [developer|commit]  $\times$  [method|method].

In the following sections we discuss the architecture of Dominoes as well as its tiles. Besides that, specialized operations that can be performed over these tiles are also described. Finally, our Dominoes GUI and its design rationale are discussed.



### 5.2.1 Architecture

Dominoes' architecture is designed in such a way that data from a software project repository is extracted and the associated change information is processed. Basically, it is composed of a set of modules responsible to extract and process data, as seen in Figure 5.1.

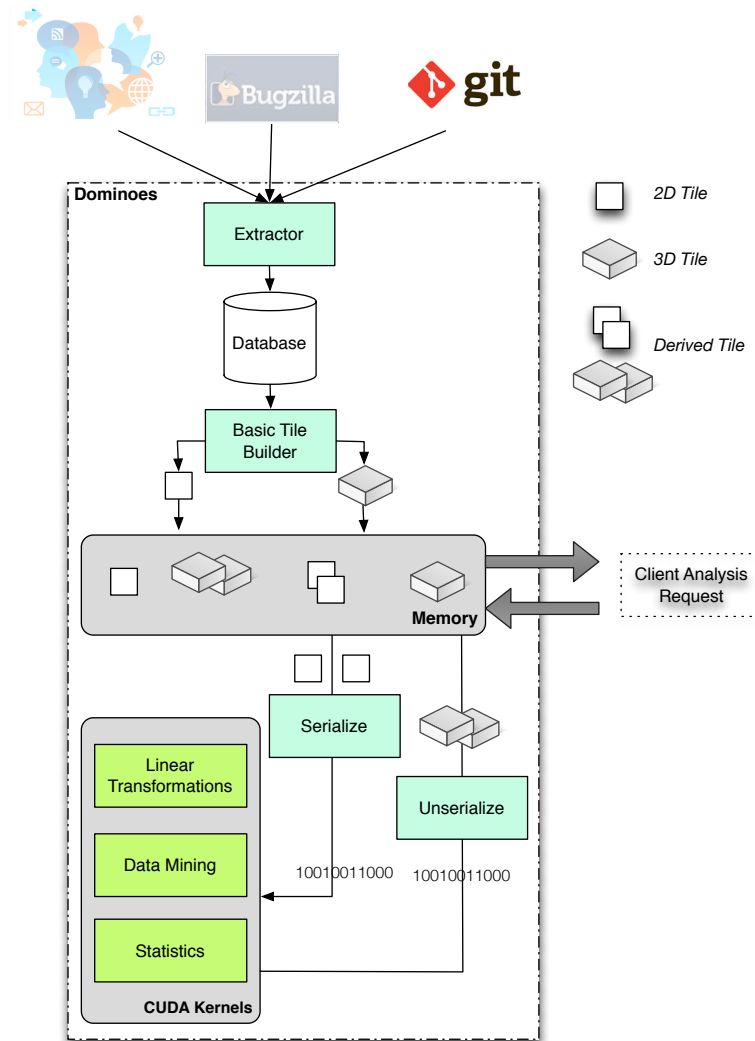


Figure 5.1: Dominoes architecture.

Currently, the **Extractor** module is responsible for mining repository projects (for version management). The repository is then preprocessed for generating a tree of all modifications performed in all commits by analyzing which files, packages, classes, and methods were modified. It is important to note that the information of each modification is decomposed to get a fine-grained view of the changes by using the Eclipse ASTParser (suitable for Java-based projects), responsible for extracting the Abstract Syntax Tree (AST). For example, even if we represent changes at the package level (for a coarse-

grained analysis), we know exactly which classes or even which methods were modified. This information is then stored in a relational database. Furthermore, after the initial data collection, information about subsequent changes can be updated incrementally to the database.

After the pre-processing stage, basic Dominoes tiles are constructed on the fly by the **Basic Tile Builder** module, which relies on querying the database in order to perform the desired relationship request, based on the granularity chosen by the user (e.g., File, Method, Package). These tiles then become available to the users, allowing them to manipulate the tiles according to their needs.

There are several additional manipulations of the data allowed by Dominoes: manipulating the set of tiles as well as filtering their values. These manipulations include **Linear Transformations** (e.g., addition, multiplication, and transposition of matrices), **Data Mining** metrics (e.g., calculating support, confidence, and lift in a tile), and **Statistics** operations (e.g., calculating the mean), as presented in Section 5.2.3. Basic building tiles can be further combined through linear transformation operations to yield derived tiles that allow exploration of derived project relationships. These derived domino tiles can also be saved as new template pieces in case that they will be used in consecutive calculations and compositions. All tiles (basic and derived) are stored in memory, allowing their use as needed for analysis since the data is cached.

Performance becomes an issue when we compute relationships at a fine-grained level. Therefore, in order to allow efficient computation at interactive speeds, we model the aforementioned manipulations as Single Instruction Multiple Thread (SMT) architecture, making possible to execute intensive matrices operations at a GPU device. When a matrix manipulation is needed, Dominoes forks its execution by triggering the respective asynchronous GPU code (*kernel*) according to the desired operation. In addition, our data is very sparse, leading to a high number of matrix' cells with zeros. In order to reduce memory consumption by just representing values different than zero, the CUSP<sup>3</sup> library is used for sparse linear algebra.

Except for the CUDA kernel operations, Dominoes is otherwise developed in Java. Performing operations over these tiles requires communicating the data with a C code, as kernels in CUDA must be programmed using the C language. Therefore, Dominoes implements a Java Native Interface (JNI) that is responsible for **serializing** and **dese-rializing** building tiles to and from C. During serialization, matrices are flattened to a

---

<sup>3</sup>Website: <https://developer.nvidia.com/cusp>

vector and submitted to GPU memory in order to be processed. After processing, the vector result is copied from GPU memory to the main CPU memory and converted back to matrices during deserialization.

### 5.2.2 Dominoes tiles

As discussed before, Dominoes includes basic building tiles, which can be combined to create derived building tiles, which can be further combined with other basic or derived tiles. The basic building tiles are created by extracting data from existing software repositories (version control systems, issue tracking systems, etc.). Due the large amount of data and size of matrices, most of them generate sparse based matrices, which will have an important optimization at the implementation. The basic building tiles around commits include:

- [class|method] (*CLM*): relationship between a class and its constituent methods, where cell  $CLM[i, j]$  has a value of 1 when class  $i$  contains method  $j$ .
- [file|class] (*FCL*): relationship between a file and its constituent classes, where cell  $FCL[i, j]$  has a value of 1 when a file  $i$  contains class  $j$ .
- [commit|file] (*CF*): relationship between commits and files modified, where cell  $CF[i, j]$  has a value of 1 when file  $j$  has been change in commit  $i$ .
- [commit|method] (*CM*): relationship between commits and methods, where cell  $CM[i, j]$  has a value of 1 when commit  $i$  adds or changes method  $j$ . Note that the index  $i$  does not necessary express the commit id.
- [developer|commit] (*DC*): relationship between developers and their commits, where cell  $DC[i, j]$  has a value of 1 when developer  $i$  is the author of commit  $j$ .
- [package|file] (*PF*): relationship between a package and its constituent files, where cell  $PF[i, j]$  has a value of 1 when a package  $i$  contains file  $j$ .
- [issue|commit] (*IC*): relationship between issue and commits, where cell  $IC[i, j]$  has a value of 1 when issue  $j$  was fixed by commit  $i$ .

These basic building tiles can then be combined to form a series of derived building tiles. In the following we show a small set of derived building tiles that can be computed using the multiplication and transposition operations:

- [method|method] ( $MM = CM^T \times CM$ ): represents method dependencies, where  $MM[i, j]$  denotes the strength of the dependency of method  $j$  on method  $i$ . The rationale of this matrix is based on logical dependencies, as elements that are co-committed together share some program logic. Note that we can also create an  $MM$  matrix through syntactic analysis, in which case it would be termed as a basic building tile. Such  $MM$  matrices have been explored by Steward [97] in Design Structure Matrices.
- [class|class] ( $CICl = CIM \times MM \times CIM^T$ ): represents class dependencies, where  $CICl[i, j]$  denotes the strength of the dependency of class  $j$  on class  $i$ . Note that using the composition tile, we can easily provide analysis results at a higher-level of abstraction.
- [issue|method] ( $IM = IC \times CM$ ): represents the methods that were changed to fix each issue. This matrix could be used to identify which methods are “buggy”.
- [developer|method] ( $DM = DC \times CM$ ): represents the methods that a developer has changed. This matrix could be used to identify experts on a particular method.
- [developer|class] ( $DCl = DM \times CIM^T$ ): represents classes that a developer has changed.  $DCl$  uses the composition operation to provide expertise information at the class level, which is typically used during bug triaging [12].
- [developer|developer] ( $DD = DM \times MM \times DM^T$ ): represents the expertise dependency among developers, where developer  $j$  depends on some knowledge of developer  $i$ , because of underlying technical dependencies in their work. Note that here this derived building tile uses other derived building tiles ( $MM$  and  $DM$ ) in its definition.

In addition, we also have the 3D tiles, which represent the whole project history and is composed of multiple slices, to be able to deal with evolution. As an example, an evolution of relationship between developers and classes across the time can be represented by a [developer|class|time] 3D tile. In this case, each slice represents a snapshot of the history at a certain moment.

A question that arises is how should we create a slice to depict the history when using 3D tiles. Humans tend to discretize evolution in terms of time; therefore, we can use time intervals (weeks, months, etc.) to discretize it. However, the project structure evolves as a sequence of commits (i.e., if no commit is performed in weeks or months,

no evolution will be perceived). We reconcile these two factors by computing a slice per unit of time (e.g., one slice per month), but in terms of a sliding window that comprises a set of commits performed before each slice. We define the size of the sliding window as presented in Equation 5.1.

$$\text{slide window size} = \text{Max}(MAM, MLC) \times M \quad (5.1)$$

Where  $MAM$  represents the number of commits in the most active month of the project history;  $MLC$  represents a minimum limit of commits per window, independently of how active is the project; and  $M$  represents a multiplier to allow users to experiment with different window sizes.

While using 3D tiles to represent evolution, we identify the number of commits that were performed in the most active month ( $MAM$ ) of a project and use that as the default size of the sliding window over which we collect commits. As previously explained we do not simply choose a month as the window size since the amount of activity in a given month can fluctuate (e.g., in open source projects) and we want to use a constant window size across our calculations. This implies that when we create slices for months that have less activity than the most active-month, they will involve commits from the previous month(s). This is in fact desirable, since having the window overlap across slices smoothens out sharp fluctuations and equally represents the effects of evolution. However, since it is possible that a (small) project might not have a month with enough activity at all to create an appropriate window size, we use a floor for the minimum number of commits ( $MNC$ ) that are used to create a slice window.

### 5.2.3 Specialized operations

Our basic matrices are typically binary, that is,  $M[i, j]$  is either 1 or 0 for any  $i$  and  $j$ , whereas our derived matrices are not. This is largely because commits are atomic transactions (i.e., a set of distinct changes are applied as a single operation) and therefore most associated matrices with commits are binary. In the case of derived matrices, cell values have associated semantics. Simple operations such as multiplication and transposition allow us to compose different types of domino tiles to derive more complex matrices and, thereby, different software engineering constructs. However, there are three “specialized” operations that can be applied on derived matrices where individual cells are not binary: support, confidence, and lift.

Let us take the example of the  $MM$  matrix. The diagonal shows how frequently a method has been changed and each cell ( $M[i, j]$ ) shows how frequently a method ( $i$ ) was changed together with another method ( $j$ ). This semantics is equivalent to the *absolute support*, largely adopted in the data mining community. The support of an item set is defined as the proportion of transactions in the dataset that contains the item set. According to [75], the rule  $X \rightarrow Y$  has support  $s$  if  $s\%$  of transactions contain  $X \cup Y$ . As this operation pattern of multiplying a matrix by its transpose is very popular and semantically rich, we treat it as a specialized operation computed according to Equation 5.2.

$$M^{sup} = M \times M^T \quad (5.2)$$

The semantics of support allows us to answer software engineering related questions regarding the strength of the relationships. For example, if we are interested in predicting which other methods a developer needs to edit because of a change, we can use the concept of logical coupling (files that are committed together have underlying logical dependencies) to identify all those methods that are dependent on the edited method and may also demand changes. We could use the  $MM = MC^{sup}$  matrix to answer this question.

Unfortunately, support is transitive, so  $M^{sup}[i, j] = M^{sup}[j, i]$ . Consequently, using support to represent dependencies is not precise, as program dependency is not transitive. In order to obtain a more precise relationship that reflects the direction of the dependency, Zimmermann et al. [104] use *confidence* to represent logical coupling. This metric suggests which artifacts should be modified together, given that a specific artifact is being modified. According to [75] the rule  $X \rightarrow Y$  has confidence  $c$  if  $c\%$  of transactions that contain  $X$  also contain  $Y$ . In the context of our approach, when applied to  $MM$  matrix, confidence quantifies the occurrence of an entity (e.g., method) change given that the other entity (e.g., method) has also changed. The confidence operator is computed according to Equation 5.3.

$$M^{conf}[i, j] = \frac{M^{sup}[i, j]}{M^{sup}[i, i]} \quad (5.3)$$

Confidence does not have a transitive property among elements, so it is possible to define different levels of dependency for each pair. However, confidence suffers from another type of problem: in the context of data mining, confidence is used to quantify

relations such as “*those who buy product A also buy product B*”. In this case, if product  $B$  is presented in almost all orders, purchase of any product will lead to a high confidence in buying  $B$ . For this reason, analyzing confidence alone tends to be imprecise, and can exhibit false relationships.

To address this problem we can use a third metric, called as *lift* [75]. Lift measures the influence of the antecedent in the frequency of the consequent. Formally, the rule  $X \rightarrow Y$  has lift  $\mathbf{L}$  if the frequency of  $Y$  increases in  $\mathbf{L}$  times when  $X$  occurs. According to this definition, we are interested in dependencies with lift greater than 1, as any other value implies irrelevant (coincidental) relationships. The lift operator is defined by Equation 5.4, where the scalar multiplication by the number of commits ( $M^{rows}$ ) transforms the absolute support ( $M^{sup}$ ) into relative support (values ranging from 0 to 1).

$$M^{lift}[i, j] = \frac{M^{conf}[i, j] \times M^{rows}}{M^{sup}[j, j]} \quad (5.4)$$

When working with some matrices, it is difficult to find a threshold for filtering out specific relationships. For instance, when using an absolute support, what is the minimum value for considering a logical coupling? To overcome this challenge, we provide the *Standard (Z) Score* (z-score) [21] to statistically identify the appropriate thresholds. It converts the absolute values (scores) into z-scores according to Equation 5.5, where  $\chi$  is the absolute score and  $\mu$  and  $\sigma$  are the mean and standard deviation of the population, respectively.

$$z = \frac{(\chi - \mu)}{\sigma} \quad (5.5)$$

## 5.2.4 Dominoes GUI

Besides Dominoes framework, we provided a graphical user interface designed to enable end users, who in our case can be project managers or developers, to perform exploratory data analysis of their projects. This section provides information about its design rationale as well as how the interface can be used.

### 5.2.4.1 Design Rationale

Dominoes’ design follows a set of guidelines that, when blended together, are able to provide a highly interactive and powerful tool for exploratory analysis of software engineering

data. In the rest of the section, we detail each of the design guidelines and their expected effects in Dominoes:

- Decouple data collection from consumption, which makes it possible to: (1) gather different types of data (version histories, issues, email, etc.), and (2) use different repositories (e.g., a choice of version control systems such as SVN, Git, Hg, etc.) from which data can be extracted. Currently, we have collected data from version histories (Git) and issue tracking systems (Bugzilla). Additional wrappers for different repositories can be implemented and “plugged” into the system.
- Support data composition, enabling users to explore different relationships and facets of data in their project to gain insight into their own projects. A key goal is to support the user in reasoning about the data relationships among project elements and starting new explorations. We did not use a query-based approach, frequently seen in the literature [44, 12] for mining relationships, because: (1) users must know or learn the query language and explicitly express how the data should be integrated, and (2) queries are often hard to formalize by end users and they are often restricted to queries that are preconceived by the tool builder during development.
- Consider data, operations, and visualizations as first-class elements, since these are the three key facets of exploratory data analysis. Dominoes allows users to interact with the data tiles to create and save more complex data units. Moreover, this separation of features facilitates extensibility by the ease in which additional operations and visualizations can be incorporated.
- Allow exploration at interactive rates. A key requirement of Dominoes is to support seamless linking of ideas and exploration of data in a “what if” fashion. This led to a collection of concrete design decisions, such as incremental data up-dates, the use of GPU for data transformation (when GPU is available), and thresholds in visualizations.
- Support open-ended explorations, allowing users to compose together different data units to make derived data types. Furthermore, we support transitioning between granularities (low to high level) by leveraging different types of composition matrices (class-method, class-package, etc.) and operations over these matrices. We also allow users to undo and redo their operations to facilitate an exploratory environment where users can view their analysis and backtrack when needed. In addition



to matrix manipulation operations (e.g., transposition and multiplication), we also allow users to aggregate (and view) data by summing across rows or columns.

- Gather provenance information over exploration sessions, allowing users to save their exploration paths in addition to saving the artifacts of their explorations (i.e., derived data types). The exploration paths can then be viewed in tree form (along with backtracks), and the user can decide whether to repeat parts of the exploration path in a later investigation or in another project. For ease of exploration we also allow users to save the workspace (the edit canvas as well as the tiles).
- Leverage visualizations. When we consider the adage that a picture is worth a thousand words, it follows that visually representing data will help users comprehend the state of their project better. Therefore, we provide a set of visualizations (graph visualization, tree structures, histograms, and also textual lists) that the user can use to display the data (the state of the matrix).

#### 5.2.4.2 Interface

We explain the different features of Dominoes by using a simple scenario. To do so, we situate the scenario in a real world project: Apache Derby.

Let us consider Mamta, who is a developer in the Derby project. She is planning to make a significant change to the code base and would like to know which other developers depend on her work. In the rest of the section, we describe how Mamta uses Dominoes to identify this information. Some steps of the process are shown in Figure 5.2, with labeled panes detailing specific actions.

Mamta starts Dominoes and is presented with the UI as shown in Figure 5.2(a). Dominoes has four panes. The first pane (top) allows users to select the time frame for analysis and presents a timeline view of project activities regarding the number of commits and the number of opened bugs. The lower left pane (library) holds all the dominoes tiles, the middle pane is the editor canvas where users can compose the tiles or operate over them. The rightmost pane is used for visualizations.

Mamta decides to use the last three months for her analysis (Nov 2013 to Jan 2014) and clicks on “Set” to start the analysis. Dominoes then generates a collection of tiles representing relationships in the project. These tiles are matrices that represent the following relationships: [commit  $\times$  method], [developer  $\times$  commit], [bugs  $\times$  commit], [class  $\times$  method], [file  $\times$  class] (see Figure 5.2(b)).

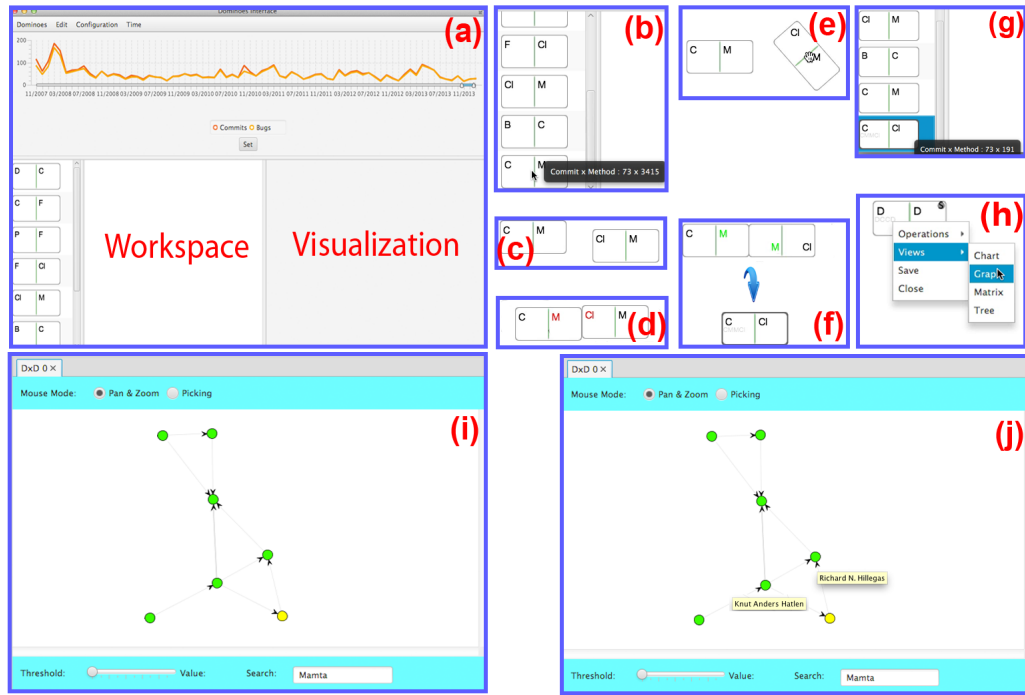


Figure 5.2: A set of panes depicting Mamta’s interactions with Dominoes. The video of the usage scenario is at: <https://github.com/gems-uff/dominoes>.

These tiles are called basic tiles [64] since they are relationships that are directly extracted from the source code, versioning system, and issue tracker. Mamta decides to start her explorations by first looking at each tile. Hovering over a tile provides information about that tile in a tool tip (Figure 5.2(b)). She decides to pick the [commit  $\times$  method] and [class  $\times$  method] tiles ([C|M], [Cl|M]). She drags these tiles from the library pane to the editor pane (Figure 5.2(c)). Since she is interested in knowing which commits were related to which classes, she wants to combine the [C|M] and [Cl|M] tiles. However, note that to combine (multiply) the matrices they need to have the same dimension (relationship). Since [C|M] and [Cl|M] do not share the same dimension, Dominoes doesn’t allow Mamta to connect these two tiles (in Figure 5.2(d) the tiles have red colored edges). Mamta realizes that to connect these two tiles she will have to transpose one of the matrices. She transposes the [Cl|M] matrix by double clicking the tile, which swivels the tile in the editor (Figure 5.2(e)). Note that she could also have right clicked on the tile and selected the “transpose” operation. Once the [Cl|M] matrix is transposed, Dominoes allows the two tiles to be combined (in Figure 5.2(f) the tiles now have green colored edges). This leads to the derived tile [C|Cl], containing information about the commits involved with the classes (Figure 5.2(f)). Mamta decides to save the derived tile for later use (Figure 5.2(g)) by right clicking on the tile and choosing the “save” option.

Mamta then continues her exploration in Dominoes. She decides to create a [commit  $\times$  commit] matrix by combining the [C|C] tile with its own transpose to generate the [C|C] tile. She then multiplies the following tiles: [D|C] ([developer  $\times$  commit]) tile, [C|C] tile, and the transpose of the [D|C] tile to generate the [developer  $\times$  developer] matrix. This matrix identifies the developers who are dependent on each other because their commits involved a common set of classes. Mamta decides to save the [D|D] tile for future use. In order to ensure that she remembers the logic behind the derived tile, since there could be other ways two developers can be related (e.g., working on the same bug), she names the tile as “dev-commit-dev”. Note that Dominoes not only allows users to save derived tiles, but also automatically saves the interaction history – the series of steps that were followed. If she wants, Mamta can replay her interaction history to generate the derived data in a new session or in another project. Note that Dominoes also allows Mamta to save her workspace, which saves all of her interactions and the tiles that she has generated.

Now that Mamta has created the underlying data she wants to identify which developers are interconnected with her. To do so, she selects the “Graph” view to visualize the [D|D] matrix (Figure 5.2(h)). The network graph (Figure 5.2(i)) shows developers (green nodes) who are interconnected because of a common set of classes that they have committed to. The view allows Mamta to set a threshold on the edge weight; that is, she can filter out developers whose edges (number of connections) are below a certain number. When she searches for her name (lower right corner of the UI in Figure 5.2(i)), Dominoes highlights the node representing her and then she can follow the edges from her node to identify the two other developers (Richard and Knut, as presented in Figure 5.2(j)) who are connected to her.

Different types of visualizations might be appropriate for different types of data. In our scenario, the network graph visualization was the easiest for discerning relationships. However, other analysis questions, such as who committed the most to a package or which files are the most buggy, might require different types of visualizations, such as bar graphs, matrices, and or tree visualizations. Dominoes currently provides these four types of visualization. Dominoes UI is built using JavaFX <sup>4</sup> whereas the visualizations are generated using JFreeChart <sup>5</sup>.

---

<sup>4</sup>Website: <http://www.oracle.com/technetwork/java/javafx/overview/index.html>

<sup>5</sup>Website: <http://www.jfree.org/jfreechart/>

## 5.3 Examples of Dominoes applicability

In this section we are going to provide examples of how Dominoes can be used for exploratory analysis. Note that this is an intentionally simple example to explain the Dominoes applicability. In addition, each class is contained in the file with the same name of the class (e.g., the class `Circle` is inside file `Circle.java`).

Initially consider a scenario where three developers (*Alice*, *Bob*, and *Carlos*) work together on a “geometry project”, containing four classes (`Circle`, `Cylinder`, `Cone`, and `Shape`). Each class has four methods but just the methods used in the following sections will be described. `Circle` has a method `circumf()` that calculates its circumference. `Shape` has a method `draw()` to render a shape. Finally both `Cylinder` and `Cone` have methods `area()` to calculate the area of the respective shapes. In addition, `Cone` has methods `collided()` to check a collision against another shape; `volume()` to calculate its volume; and `copy()` to make a copy of itself.

Using the previous established scenario, the following sections introduce how to use Dominoes for calculating dependencies as well as showing how Dominoes can be used for expertise identification.

### 5.3.1 Calculating dependencies using Dominoes

Using the established scenario, this section shows how data mining concepts such as support, confidence, and lift matter when identifying artifact dependencies.

In order to begin the discussion, Table 5.1 describes five commits and their change descriptions. Table 5.2 shows which commits modified which methods.

Table 5.1: Commits made by developers.

Commit #	Developer	Description
C1	Alice	Change type of function parameter to compute the radius ( <code>Circle</code> ) and how to render it (in <code>Shape</code> )
C2	Carlos	Change the side of <code>Cone</code> and how to render it
C3	Alice	Change how a <code>Shape</code> is rendered
C4	Alice	Calculation of how circumference and area are calculated using $\pi$ . Required modification on how to draw a <code>Shape</code>
C5	Bob	Modify the height calculation of a cylinder and how it is rendered

Table 5.2: Methods changed for commit.

Commit #	Circle circumf()	Cylinder area()	Cone area()	Shape draw()
C1	1	1	0	1
C2	0	0	1	1
C3	0	0	0	1
C4	1	1	1	1
C5	0	1	0	1

Figure 5.3 represents the support, confidence, and lift values for the *MM* matrix, where Ci represents *Circle.circumference()*, Cy - *Cylinder.area()*, Co - *Cone.area()*, and S - *Shape.draw()*.

	Ci	Cy	Co	S		Ci	Cy	Co	S		Ci	Cy	Co	S
Ci	2	2	1	2	Ci	1	1	0.5	1	Ci	2.5	1.6	1.2	1
Cy	2	3	1	3	Cy	0.6	1	0.3	1	Cy	1.6	1.6	0.7	1
Co	1	1	2	2	Co	0.5	0.5	1	1	Co	1.2	0.8	2.5	1
S	2	3	2	5	S	0.4	0.6	0.4	1	S	1	1	1	1
	Support					Confidence					Lift			

Figure 5.3: Support, Confidence, and Lift calculated from previous scenario.

If we consider the confidence matrix, we notice that the dependency from Cy to Ci (100% confidence - row 1 column 2) is stronger than from Ci to Cy (60% confidence, row 2, column 1), because whenever Ci was changed, Cy was also changed (commits C1 and C4) (see Table 5.2). However, Cy was changed once without Ci (commit C5). With such a confidence analysis we can state that Cy (always) depends on Ci, but Ci does not necessarily depend on Cy. Therefore, using confidence to derive the *DD* matrix would identify that Bob should communicate with Alice, but not necessary the opposite way.

The confidence matrix also indicates high dependency from S to all other methods. However, this occurs not because S really depends on all other methods, but because S was independently changed in all commits (see Table 5.2). The lift matrix eliminates such coincidental dependencies, keeping only dependencies between Cy and Ci, and Co and Ci, since all other values are either equal to or below 1.

In summary, support alone is not sufficient to indicate dependencies among project entities, but helps in eliminating dependencies that appear by chance (e.g., Co and Ci). In a large project, with thousands of commits, thresholding on a predefined support level can help to eliminate accidental dependencies. On the other hand, lift plays a complementary role of identifying dependencies to elements that are very frequent (e.g., S) and, therefore, avoiding coincidental changes. Finally, confidence is important to identify the direction

of the dependency (e.g., from  $C_y$  to  $C_i$ ). With such an analysis, we find that the only real dependency in our scenario is from  $C_y$  to  $C_i$ , which would lead to a communication requirement from Bob to Alice in the  $DD$  matrix.

Our approach, therefore, provides four distinct advantages. First, the confidence measure allows more nuanced investigations (e.g., direction of dependency). Second, the use of lift measure coincidental relationships but unrelated changes. Third, the fine-grained analysis at the method level increases accuracy, since we can identify dependencies among individual methods. Therefore, if we find that  $C_y$  depends on  $C_i$ , we can find that Bob needs to coordinate with Alice, who is working on  $C_i$ , and not another developer who is working on the same file (Circle, but on a different method). Finally, GPU processing allows these investigations to be performed interactively.

### 5.3.2 Expertise identification using Dominoes

In this section, we introduce how Dominoes can be used for identifying expertise of developers over artifacts in a project. It considers two important factors pertaining to expertise: granularity of analysis and time. In essence, artifacts are not atomic, and local expertise (in specific parts of an artifact) should be differentiated from global expertise (in the whole artifact). Moreover, as artifacts naturally evolve over time the expertise of a developer diminishes unless she has kept familiarity with the artifacts through time. Initially, the importance of granularity during analysis is being discussed. Later, we analyze how this expertise varies across time.

#### 5.3.2.1 Granularity Matters

In this subsection we discuss two different strategies for calculating the expertise of a developer in a given artifact (e.g., file): Expertise of a Developer (ED) and Expertise Breadth of a Developer (EBD). The first one considers the entire artifact as an atomic element. This strategy is vastly adopted in the literature [82, 22, 93]. However, to differentiate among developer expertise, we use the z-score in our analysis. The second approach uses the underlying composition structure of the artifact (e.g., methods) to perform a fine-grained analysis when calculating expertise.

**Expertise of a Developer (ED):** it identifies the frequency of changes to an artifact (e.g., file, project, etc.) by a given developer. Frequency of edits has long been used as a proxy for identifying the knowledge that a developer has about an artifact, typically a file

[5]. The intuition is that the more someone has edited a file, the more working knowledge that person has about that file. The frequency of edits can therefore help answer two related questions: (1) “who is the developer that is an expert for a given file?”, and (2) “which files is a given developer an expert of?”.

Table 5.3 shows the previous established scenario after a set of commits. Based on this scenario, it is possible to build a tile relating the three developers who have worked on four files ([developer|file] - DF for short). Note that we arrive at the DF matrix by operating over the basic tiles ([developer|commit]  $\times$  [commit|file]). The cells in the derived matrix DF represent the number of occurrences (e.g., the number of times a developer  $d_i$  edited a file  $f_j$ ), which is considered as “support” in data mining nomenclature. Besides that, Table 5.3 also shows the number of commits performed by each developer (note that it is different from summing all columns in a row, as a commit may comprise more than one file).

Table 5.3: Developer  $\times$  File.

<i>Project</i>	<i>Circle.java</i>	<i>Cylinder.java</i>	<i>Cone.java</i>	<i>Shape.java</i>	<i># Commits</i>
<i>Alice</i>	14	2	20	1	28
<i>Carlos</i>	10	24	12	1	25
<i>Bob</i>	25	10	8	4	40

To answer the first expertise question (who is an expert for a given file  $f_j$ ), we search for the developers who mostly edited the file. This is done by scanning down the column of  $f_j$  in the DF matrix. In our simplistic example (see Table 5.3), if we want to identify an expert for *Cone.java*, we would indicate Alice. Carlos would be considered as the second most knowledgeable developer in that file.

To answer our second question, about the expertise of a specific developer  $d_i$ , we scan the rows in the matrix for the highest values. In our example, we find that Alice has expertise in *Cone.java*, Carlos in *Cylinder.java*, and Bob in both *Circle.java* and *Shape.java*.

A key challenge is identifying the right threshold to use in this approach. For example, “is there a minimum number of changes that developers must have performed before they can be considered as experts?”. Further, if two developers have changed the file, how do we determine who can be defined as “the” expert - should that be the person with the most edits? For example, if we are to compare the expertise of Alice versus Carlos on *Circle.java*. Because Alice has made four more changes than Carlos, is it possible to conclude that Alice clearly have more expertise than Carlos? Does a difference of four

additional edits matter, or should there be a minimum distance between the numbers of edits by developers to differentiate expertise between developers?

To overcome this challenge, we applied the z-score, where  $\chi$  is the absolute score in Table 5.3 and  $\mu$  and  $\sigma$  are the mean and standard deviation of the number of edits for each file, respectively. When using z-scores (see Table 5.4), cells above zero indicate values that are above the mean, that is, they indicate that a developer has changed that file more than the mean number of times that the file has been changed in the project. Similarly, cells below zero indicate values below the mean. Moreover, cells above (or below) one, two, or three indicate values above (or below) one, two, or three standard deviations from the mean, respectively. For example, when we consider *Cone.java*, we see that it has been changed on average 13.33 times (summing column 4 in Table 5.3 and dividing by the total number of developers in the project), and that Alice has edited the file 1.34 standard deviations above the mean (Table 5.4, cell value for Alice’s edits to *Cone.java*).

Table 5.4: Standard Score.

<i>Project</i>	<i>Circle.java</i>	<i>Cylinder.java</i>	<i>Cone.java</i>	<i>Shape.java</i>
<i>Alice</i>	-0.37	-1.10	1.34	-0.71
<i>Carlos</i>	-1.00	1.32	-0.27	-0.71
<i>Bob</i>	1.37	-0.22	-1.07	1.41

We assume zero as the threshold for determining expertise. That is, to be counted as an expert of a file a developer must have edited more than the mean change rate of that file. In our example, this measure enables us to quickly identify the expert developer for each of the files. Alice can be considered an expert in *Cone.java*, since she has made significantly more changes than any other developer in the project. Similarly, Bob is an expert in both *Circle.java* and *Shape.java*, and Carlos in *Cylinder.java*. Similarly, to be considered a higher expert than another we assume that the developer has to present edits that is one standard deviation higher.

**Expertise Breadth of a Developer (EBD):** here we challenge the assumption that a developer who has more commits over a file always has the most (breadth) knowledge of this file. Normally, a file comprises of a set of classes and methods and it is possible that a developer only performs niche changes to a sub-set of methods or internal classes. In such cases, numerous edits to only a subpart of the file do not guarantee that the developer is an expert on the entire file. We, therefore, analyze changes at different levels of granularity to create a more nuanced understanding of expertise.

To calculate EBD we run an analysis at a fine grain. In our case study we run an



analysis at the method level, however, it is also possible to work at the lines-of-code level if needed. We first calculate the absolute scores (count of number of edits to the [developer|method] matrix, DM for short, which is computed as [developer|commit]  $\times$  [commit|methods]). We then transform the absolute scores into z-score. Finally, using zero as a threshold, we count the cells that have positive numbers as a measure of expertise.

As an example, consider Table 5.5 (DM matrix), which is composed of methods present in *Cone.java* for class *Cone* and the number of commits performed by each developer, which involved those methods. Table 5.6 presents the z-score calculated from Table 5.5. In this example, it is possible to see that even though Alice has edited *Cone.java* the most (20 times) and therefore had the highest ED for this file (Table 5.3), most changes she has made were only related to a single method in the class of this file. In contrast, Carlos has made only 12 changes (as compared to 20 by Alice), but his changes are spread across all the methods in the class of the file. Therefore, based on our proposed metric, Carlos has a higher EBD in *Cone.java* (in Table 5.6 Carlos has positive entries for three methods). If we now look at the expertise of developers in the project that comprises these four files, we find that Carlos is the expert in both *Cone.java* and *Shape.java*, and Alice in *Circle.java*. Carlos and Bob both have expertise in *Cylinder.java* (see Table 5.7, which presents experts by each file as well as its z-score in parentheses; we use 0 as threshold when determining expertise).

Table 5.5: Developer  $\times$  Method (DM matrix).

<i>Cone.java</i>	<i>area()</i>	<i>collided()</i>	<i>volume()</i>	<i>copy()</i>
<i>Alice</i>	2	1	1	16
<i>Carlos</i>	4	3	4	1
<i>Bob</i>	2	1	3	2

Table 5.6: Developer  $\times$  Method z-score.

<i>Cone.java</i>	<i>area()</i>	<i>collided()</i>	<i>volume()</i>	<i>copy()</i>
<i>Alice</i>	-0.71	-0.71	-1.34	1.41
<i>Carlos</i>	1.41	1.41	1.07	-0.78
<i>Bob</i>	-0.71	-0.71	0.26	-0.63

Finally, calculating EBD for the project as a whole when considering the method-grain based approach in Table 5.7 (note that here we omit presenting how the z-score at method level for *Circle.java*, *Cylinder.java*, and *Shape.java* has been derived for easing visualization) reveals that Carlos has the highest level in the project (he has three cells that have positive z-scores) when compared to Alice and Bob (both have only one cell

with positive z-score). Therefore, EBD can serve as a more precise measure of identifying expertise of a developer when fine-grained data is available, instead of simply using ED.

Table 5.7: Expertise and z-score at file level.

<i>Project</i>	<i>Circle.java</i>	<i>Cylinder.java</i>	<i>Cone.java</i>	<i>Shape.java</i>
<i>Alice</i>	3 (1.41)	0 (-1.34)	1 (-0.71)	0 (-1.07)
<i>Carlos</i>	1 (-0.71)	3 (1.07)	3 (1.41)	3 (1.34)
<i>Bob</i>	1 (-0.71)	2 (0.27)	1 (-0.71)	1 (-0.27)

Recall that Dominoes can perform analysis at different levels of granularity by using the appropriate composition matrix. Therefore, it can be used to indicate expertise by analyzing edits at different levels of granularity. Here we explained how expertise of developers can be calculated for the project by recursively considering edits from the method level (called  $EBD^M$ ) to the file level and then to the project level. That is, we give expertise credit to a developer for a file if she has a z-score above zero when considering changes to methods of that file. The intuition here is that a developer is considered an expert in a file if she has breadth of knowledge in that file (z-score  $> 0$ ), and this is recursively computed for the project.

We can follow a similar strategy to identify experts for the project by analyzing edits directly from the file level (i.e., we give expertise credit to a developer for each file, in which she has z-score of commits above zero for that file). We call this  $EBD^F$ . The intuition here is that to be considered an expert, a developer must have changed a file more than average, and that information is aggregated for the project. The only difference is that the latter considers files as atomic elements and does not take into account edits to their methods.

Considering our example, if we simply used ED for determining expertise for the project, that is, the person who has made most changes to the project as the highest expert, then Bob would be considered the expert, followed by Alice and Carlos (see Table 5.3). When considering  $EBD^F$ , we see that Bob would still be considered as the highest expert followed by Alice and Carlos, as counting the number of files modified above the mean is equal two for Bob and one for both Alice and Carlos in Table 5.4. However, when using  $EBD^M$ , we notice that Carlos has the highest expertise, as he is the expert in three out of four files, while the others are expert in only one out of four files, as shown in Table 5.7.

### 5.3.2.2 Time matters

As a project evolves, it is possible that developers take on different roles or move to different parts of a project. Typical archival analyses for expertise identification do not take project evolution into consideration [82, 23]. As a result of this, developers who had made frequent changes in the past, but are no longer active and unaware of the current project structure, may still be recommended as experts.

In our illustrative example, performing expertise analysis over such slices might show that Bob has the highest expertise in *Shape.java* (see layer 1 in Figure 5.4), but as we visit subsequent slices we see that the expertise shifts in time, with Alice being the expert in the last window (layer 3 in the Figure 5.4) of the project. Such a time-based analysis can show when an expertise handover occurs in a project (e.g., between periods 2 and 3 Bob makes much fewer edits to the *Shape.java* and Alice assumes development for that file). Such analyses can, therefore, identify points when a developer started acquiring expertise, how long it took to gain expertise to the extent of the previous expert, and for how long a developer’s expertise is valid (as software changes, past expertise naturally loses strength). Therefore, Dominoes allows more nuanced investigation of expertise - for example, it can help answer how expertise has changed over time.

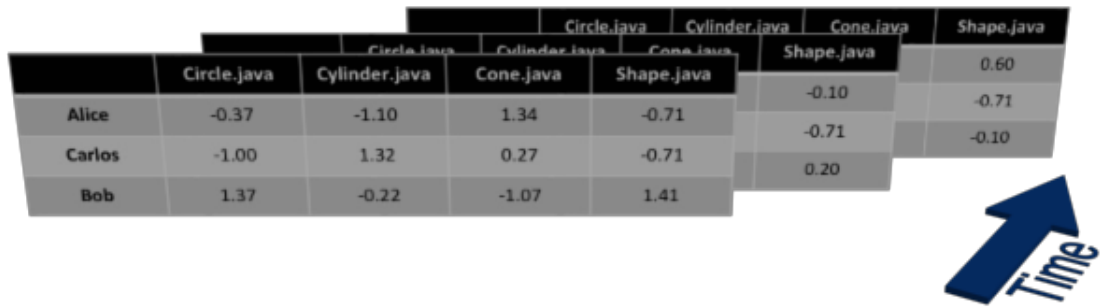


Figure 5.4: [developer|file|time] tile with layers in the back denoting recency.

## 5.4 Case studies

In this section we report on the evaluation of Dominoes in the context of dependency calculation, expertise identification and its evolution, and performance evaluation. We used the open source project Derby as our experimental object. We selected Apache Derby because it has a large, active contributor base and has a good number of commit-issue linkages (about 80%). All analyses were performed over the data extracted from

Derby repository from Aug 2004 to Jan 2014, which comprises **7,573** commits, **36** unique developers, **34,335** file changes, and **305,551** method changes committed during a span of about 10 years. Additionally, for all the experiments a CPU (Intel Core 2 Quad Q6600) equipped with a GPU (NVidia GeForce GTX 580) was used.

### 5.4.1 Dependency identification

We first created the *MM* matrix (dependencies between methods based on co-commit information) and then applied the composition operation to calculate the *ClCl* (class to class) matrix. We found that in Derby a file was associated with only one class, therefore, for our purposes the *ClCl* matrix is the same as the *FF* ([file|file]) matrix. We then applied the confidence and lift analysis on this matrix. We found that due to the characteristics of Derby, the lift analysis does not filter out any coincidental dependencies. This is because the Derby file dependencies are highly clustered, causing low support and a very high lift. When we filter the lift values by thresholding it with “1”, no data points were eliminated. Therefore, here we restrict our discussion to the support and confidence analysis.

Table 5.8 presents the top 5 logical dependencies in terms of support and with the biggest difference in confidence (the filename extension is omitted). It is important to remember that confidence is not transitive.

Table 5.8: Top 5 logical dependencies in terms of high support and biggest confidence difference.

Artifact A	Artifact B	Support	Conf. (A-B)	Conf. (B-A)
DataDictionary	DataDictionaryImpl	79	0.88	0.37
DD_Version	DataDictionaryImpl	45	0.78	0.21
LanguageConnection Context	GenericLanguageConnection Context	44	0.86	0.48
DRDAConnThread	DRDAStatement	37	0.22	0.68
ResultSetNode	SelectNode	36	0.54	0.45

Considering the first case as an example, it is possible to observe that using the common approach that is based on support, artifacts *DataDictionary.java* and *DataDictionaryImpl.java* would be considered as dependent on each other as they have a high support (in fact, 79 is the highest value of absolute support in the whole system). However, when observing the confidence, it is possible to see that only *DataDictionaryImpl.java* has dependency with *DataDictionary.java*, which is reasonable as changing a method implementation normally does not result in a change to its interface. The follow-

ing two rows are also interface/implementation cases, presenting the same behavior. In the fourth row, we have a composition case, where *DRDAConnThread.java* possesses a *DRDAStatement.java* instance. In this case, modifying the former does not necessary imply a modifications in the latter. However, there is a high likelihood of a related change in the opposite direction, that is, modifications in *DRDAStatement.java* can change method signatures used by *DRDAConnThread.java*, for instance.

Finally, the last case is a class specialization, which normally requires modification to both files, with a slightly higher dependence from the subclass to the superclass. These analyses show the importance of using confidence to identify the direction of the dependencies.

Besides these five top dependencies, Figure 5.5 presents a scatter plot chart with all dependencies at three specific support levels (10, 20, and 30). This chart plots each dependency according to its confidence in both directions (A-B and B-A). This way, dependencies with the same confidence value in both directions are plotted along the diagonal. As we can see, however, there are several cases where points are located far from the diagonal. When we consider the rightmost chart in Figure 5.5 (support threshold at 30) for discussion, we can observe some distinct patterns. The red quadrant shows that both **conf(A-B)** and **conf(B-A)** are less than 0.5, thus containing weak bidirectional dependencies. The yellow quadrant, on the other hand, shows dependencies where both **conf(A-B)** and **conf(B-A)** are above 0.5, thus containing strong bidirectional dependencies. Finally, the green and blue quadrants show unidirectional dependencies with highest divergences among confidence. In this case, dependencies from these quadrants can be erroneously classified as bidirectional if we solely use support to analyze dependencies.

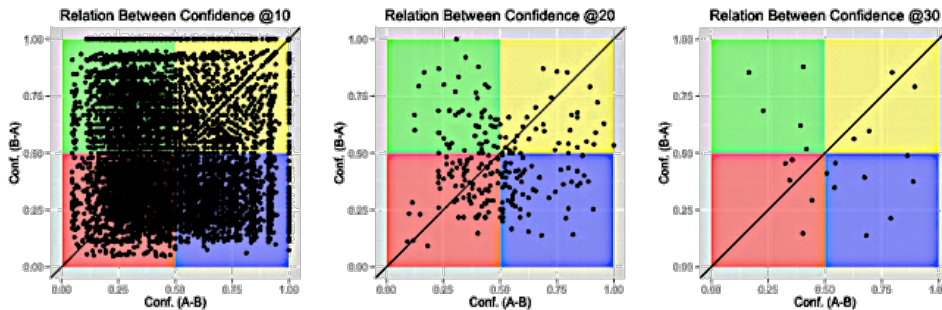


Figure 5.5: Relation among confidence for various support threshold. The leftmost chart considers a threshold of 10, while the middle uses 20, and finally the rightmost uses 30.

Performing an analysis such as the one illustrated in Figure 5.5 can unveil how inaccurate dependencies extracted from support-based approaches tend to be. As demonstrated

for the Derby project, only dependencies in the yellow-quadrant should be classified as bidirectional. Both blue and green quadrants present unidirectional dependencies.

### 5.4.2 Expertise identification

In this section, we contrast analyses using fine-grain versus coarse-grain data when identifying expertise in the Derby project by: (1) computing ED and EBD metrics for each file and (2) comparing the experts for each file based on these metrics. ED and EBD metrics diverged in **977** files, totalizing in **28%** difference in expertise calculations. As previously discussed, this difference is a result of how ED is calculated: summing up the number of times a developer edited a file. Because of this, even if a developer has worked on only very specific parts of the file, she is considered an expert for the entire file. This is how expertise is normally identified by current approaches [82, 22, 93], potentially leading to imprecision in the recommendations.

To demonstrate the difference when we calculate expertise at the coarse-grain (ED) versus at the fine-grain (EBD), let us consider the file “*CreateAliasConstantAction.java*” in the Derby project. This file comprises only one class with four methods: (1) *CreateAliasConstantAction()*, (2) *executeConstantAction()*, (3) *toString()*, and (4) *vetRoutine()*. Table 5.9 presents the expertise (ED) of two developers who edited this file the most. The mean (1.88) and standard deviation (1.91) were computed out of 17 commits made by 9 developers who edited the file in total.

On the other hand, Table 5.10 presents the expertise breadth (EBD) of the same two developers from Table 5.9 as well as the number of commits, mean, and standard deviation in order to highlight the difference between ED and EBD. It is important to note that ED basically represents the number of commits and z-score of each developer over the file. For instance, 7 commits are equivalent to 2.67 standard deviations above the mean and 3 commits are equivalent of 0.58 standard deviations above the mean.

Table 5.9: Absolute and z-score ED for *CreateAliasConstantAction.java*.

<i>Expertise of a Developer (ED)</i>		
	<i>Absolute ED</i>	<i>Z-Score ED</i>
<i>djd</i>	7	2.67
<i>rhillegas</i>	3	0.58

When we calculate expertise based on the number of commits that each developer made to this file, we see that *djd* has a higher expertise (ED) when compared to *rhillegas*. However, when using EBD we find that *rhillegas* has a higher expertise breadth in the

Table 5.10: Absolute and z-score EBD for CreateAliasConstantAction.java.

<i>Expertise Breadth of a Developer (EBD)</i>					
<i>Developer</i>	<i>Method 1</i>	<i>Method 2</i>	<i>Method 3</i>	<i>Method 4</i>	<i>EBD</i>
<i>djd</i>	1 (-0.57)	2 (0.30)	1 (-0.57)	0 (-0.16)	1 (-1.00)
<i>rhillegas</i>	2 (1.73)	3 (1.50)	2 (1.73)	1 (1.00)	4 (1.00)
<i>Total Commits</i>	5	7	5	1	-
<i>Mean / St. Dev.</i>	1.25 / 0.43	1.75 / 0.89	1.25 / 0.43	1.0 / -	-

file. The numbered columns show the absolute and z-score (in parentheses) commits for each method. The last column shows the absolute and z-score (in parentheses) EBD values for the whole file. As previously discussed, the absolute EBD at file level is the number of methods each developer has modified above the mean (above zero z-score). In this case, it is possible to see that *djd* has modified just one method above the mean (*CreateAliasConstantAction()*), while *rhillegas* has modified all of the methods above the mean. Consequently, we can consider that *rhillegas* (EBD value of 4) has a wider knowledge over this file than *djd* (EBD value of 1).

It is important to note that, because we are using a fine-grained approach for calculating EBD, we are able to distinguish between modifications that change the method body, as compared to modifications that do not affect methods (e.g., inserting comments or import statements). For instance, in our example we find that *djd* has committed to the file seven times. However, only four of the changes really affected the methods. On the other hand, *rhillegas* committed the file three times, but each commit changed more than one method (e.g., Method 2 was modified by all three commits). Therefore, when analyzing commits per file, *djd* would be considered an expert, whereas using the fine-grained EBD view, we see that *rhillegas* has a broader expertise. In fact, *rhillegas* has changed each individual method more times than *djd* (i.e., *rhillegas* dominates *djd* in all methods).

Next, we calculate expertise for the file "*EmbedConnection.java*", since it is a large file (it is comprised by 135 methods) and has been edited extensively. Table 5.11 shows the difference when comparing ED and EBD for this file. From this table, we see that *djd* has the most edits to the file (ED). However, he only touched 18 methods above the mean (EBD<sup>M</sup>), leading to a low z-score. In contrast, *kristwaa* committed to this file only 8 times (ED), but her modifications touched 22 methods above the mean (EBD), leading to a higher z-score when compared to *djd*. We find that *rhillegas* has the highest expertise since he edited 59 methods above the mean.

Finally, *bpendleton* appears in Table 5.11 with two commits in the ED list. However,

Table 5.11: Top expert developers at file *EmbedConnection.java* by ED and EBD<sup>M</sup>.

<i>Dev.</i>	<i>ED</i>	<i>Z-score</i>	<i>Dev.</i>	<i>EBD<sup>M</sup></i> ( <i>Method Level</i> )	
<i>djd</i>	21	2.28	<i>rhillegas</i>	59	2.04
<i>kahatlen</i>	19	1.98	<b><i>coar</i></b>	56	1.89
<i>rhillegas</i>	16	1.53	<i>kahatlen</i>	50	1.59
<i>oysteing</i>	14	1.23	<i>kristwaa</i>	22	0.21
<i>dag</i>	10	0.63	<i>dag</i>	22	0.01
<i>kristwaa</i>	8	0.33	<i>djd</i>	18	0.01
<i>kmarsden</i>	6	0.03	<i>oysteing</i>	18	0.01
<i>abrown</i>	2	-0.56	<i>kmarsden</i>	8	-0.47
<b><i>bpendleton</i></b>	2	-0.56	<i>bernt</i>	4	-0.67
<i>lilywei</i>	2	-0.56	<i>bandaram</i>	2	-0.77
<i>bandaram</i>	2	-0.56	<i>lilywei</i>	2	-0.77
<i>bernt</i>	1	-0.71	<i>tmnk</i>	1	-0.82
<i>davidvc</i>	1	-0.71	<i>suresht</i>	1	-0.82
<i>dyre</i>	1	-0.71	<i>abrown</i>	1	-0.82
<b><i>coar</i></b>	1	-0.71	<i>mamta</i>	1	-0.82
<i>mamta</i>	1	-0.71	<i>dyre</i>	1	-0.82
<i>bakksjo</i>	1	-0.71			
<i>suresht</i>	1	-0.71			
<i>tmnk</i>	1	-0.71			

one of his commit was made because of importing modifications (not considered when calculating EBD), and the other commit touched just one method. Therefore, he does not appear in the EBD list. In contrast, ***coar***, who has just one commit (15<sup>th</sup> position in the ED list), has modifications to 56 methods above the mean (2<sup>nd</sup> position in the EBD list) since he was responsible for the initial code creation.

The same approach can be applied at the project level to identify expertise when considering the entire project history. There are three ways in which we can calculate this, as discussed in Section 5.3.2.1. First, we calculate the total number of commits performed by each developer in the project (ED column) as shown in Table 5.12. This metric follows the intuition that the more commits a developer has made to a project, the more knowledge he has about the project.

Second, we calculate EBD for the project by aggregating changes at the file level (i.e., we give expertise credit to a developer for each file, in which she has Z-score of commits above zero for that file), called EBD<sup>F</sup>. Finally, we calculate EBD for the project by aggregating changes from the method level (EBD<sup>M</sup>). That is, we give expertise credit to a developer for a file if she has a Z-score above zero when considering changes to methods of that file.



Table 5.12: Top 10 expert developers by ED,  $EBD^F$ , and  $EBD^M$ .

<i>Dev.</i>	<i>ED</i>	<i>Dev.</i>	<i>EBD<sup>F</sup></i> ( <i>File Level</i> )		<i>Dev.</i>	<i>EBD<sup>M</sup></i> ( <i>Method Level</i> )	
<a href="#">kahatlen</a>	1393	djd	1846	2.61	djd	1533	3.42
djd	1190	rhillegas	1780	2.49	<b>coar</b>	<b>1244</b>	<b>2.66</b>
rhillegas	990	<a href="#">kahatlen</a>	<a href="#">1556</a>	<a href="#">2.08</a>	<a href="#">kahatlen</a>	<a href="#">1047</a>	<a href="#">2.13</a>
kmarsden	650	<a href="#">bakksjo</a>	<a href="#">1252</a>	<a href="#">1.52</a>	rhillegas	960	1.90
kristwaa	640	<b>coar</b>	<b>1210</b>	<b>1.45</b>	<b>dag</b>	<b>631</b>	<b>1.03</b>
fuzzylogic	412	kristwaa	1045	1.15	kmarsden	467	0.60
myrnavl	385	fuzzylogic	893	0.87	kristwaa	403	0.43
<b>dag</b>	<b>331</b>	<b>dag</b>	<b>879</b>	<b>0.84</b>	<a href="#">bandaran</a>	<a href="#">361</a>	<a href="#">0.32</a>
<b>mikem</b>	<b>284</b>	<a href="#">davidvc</a>	<a href="#">866</a>	<a href="#">0.82</a>	fuzzylogic	322	0.21
mamta	282	bpendletc	823	0.74	bpendletc	245	0.01

When we compute the top 10 experts in the project, we see that *kahatlen* appears as the highest expert in the project when considering the ED measurement. However, in  $EBD^F$  and  $EBD^M$  he moves to the 3<sup>rd</sup> position. Conversely, *djd* assumes the 1<sup>st</sup> position for both  $EBD^F$  and  $EBD^M$  measures, which means that *djd* has more breadth in knowledge across the whole project than *kahatlen*. This clearly shows that calculating expertise by simply computing the number of commits performed by a developer can yield to very different results. Moreover, *coar*, who appears in the 5<sup>th</sup> and 2<sup>nd</sup> positions in  $EBD^F$  and  $EBD^M$ , respectively, does not even appear in the ED list. The same situation occurs with *dag*, who moves from the 8<sup>th</sup> position according to both ED and  $EBD^F$  to the 5<sup>th</sup> position when considering  $EBD^M$ . Also note that *bandaran* (colored in green) in the  $EBD^M$  list does not appear in either ED or  $EBD^F$  lists. On the other hand, two other developers in  $EBD^F$  list (*bakksjo* and *davidvc*, colored in red) do not appear in the  $EBD^M$  list. This shows that many developers make edits to only some portions of the file and may not have expertise over the majority of the methods in a file.

Another interesting case occurs when we consider *bakksjo*, who is at the 4<sup>th</sup> position in the  $EBD^F$  list with a z-score of 1.52. When considering  $EBD^M$ , he falls down to the 26<sup>th</sup> position in this list (not shown in Table 5.12), presenting a value of 8 and a z-score of -0.61. This happens because *bakksjo* has modified over 1300 files. However, most of his changes introduced comments and copyright modifications, without any functionality changes in the code. This is another example of how simply calculating edits to a file may not be the right approach to identifying expertise on that file. These results show how fine-grained analyses at the method level can provide a more nuanced view of expertise. Consequently, another advantage of analyzing edits at the method level is the ability to filter out non-code related changes. Previous research has used commit size as a mechanism to filter out

copyright and other non-code related changes [79, 46]. Our results show that analyzing edits at the method level provides a more accurate means of filtering out these ancillary commits.

### 5.4.3 Expertise evolution

Another important aspect to consider when analyzing expertise is time. As software and people change over time, it is expected that expertise also evolve. The Apache Derby project, which is a long living and active project with a 10-year history, is not an exception. We analyze expertise evolution through two studies, which identified changes in expertise: (1) in one particular file, and (2) in the entire project. For both analyses we group commits by using Equation 5.1.

We selected the file *"EmbedConnection.java"* to measure expertise evolution when considering a single file, because it was the third most committed file in the project and was “touched” by a large number of developers. When calculating the size of the sliding window for the analysis (see Equation 5.1), MAM is equal to 11, as this was the number of commits that occurred in the most active month, which was Feb 2008. MNC equals to 10, as we believe this provides a sufficiently large slice of data. Note that our choice of MNC does not influence the results, as it is lower than MAM. We did not change the M factor; therefore, our sliding window size is 11.

We derived the [developer|method] tiles for each layer, based on the aforementioned sliding window, and stacked them to create a [developer|method|time] 3D tile for *"EmbedConnection.java"*. We then computed a time-based EBD for it.

Figure 5.6 shows the graph of EBD evolution for each developer who have edited *EmbedConnection.java*. We observe that *djd* was the most active developer for almost half a year. Then, onwards *kahatlen* assumed expertise for almost a year, followed by *rhillegas*, both relinquishing their expertise at the same time. The graph shows that *rhillegas*, after almost three years since the beginning, starts to become an expert again in this file. A few months after this, *oysteing's* expertise starts to decline. At the end, *rhillegas* can be considered the expert in this file. Note that if we were to determine an expert for the file by using ED (and over the entire project) *djd* would be considered the expert (Table 5.11). On the other hand, when we consider the evolution of expertise (using EBD), we find that *rhillegas* has the most expertise breadth at the end.

For the second study (temporal analysis of expertise across the project as a whole),

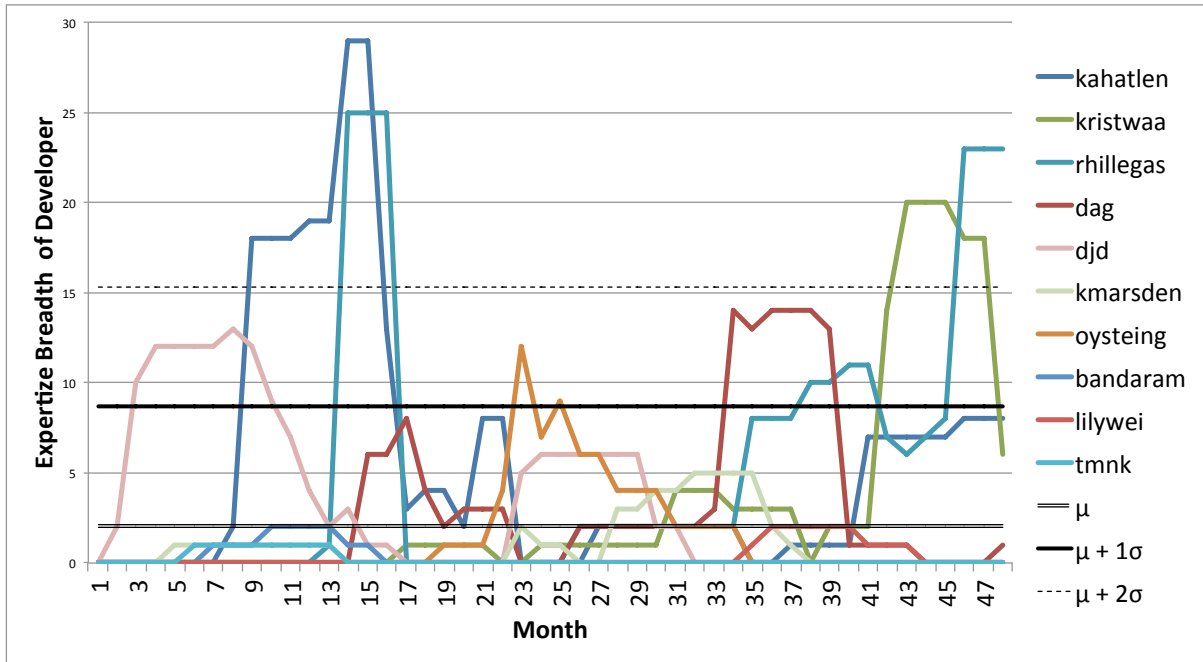


Figure 5.6: Developer breadth expertise for file *EmbedConnection.java*.

when we consider the sliding window for analysis we group commits by setting MAM to 270 (Equation 5.1). This was the maximum number of commits that occurred in the most active month (Aug 2006). We set MNC to 100, as we believe this provides a sufficiently large slice of data over which to perform analysis. Again, our choice of MNC does not influence the results, as it is lower than MAM. Also, the M factor was not changed, leaving the sliding window size to 270.

We then compute EBD for the entire project by considering each time slice. Here we show the results of our analysis of the top five developers in terms of EBD in Table 5.12.

Figure 5.7 shows the evolution of EBD of these developers. In this graph we find that *djd* was the main developer in the beginning of the project, together with *coar*. Almost a year and a half later, *rhillegas* and *kahatlen* start to contribute more extensively to the project, increasing their expertise. Four years after the start of the project, *djd* leaves the project and *rhillegas* takes over the development. In the fifth year of the project *rhillegas* has his first month with EBD higher than two standard deviations above the mean (higher dashed line). While *rhillegas* still remains very active, after six years *kahatlen* seems to supersede *rhillegas* in terms of EBD. This clearly contrasts with the results presented in Table 5.12: if we do not consider evolution, *djd* would be considered the main expert.

Finally, *dag* appears as the leading expert in Derby's recent years (superseding *kahatlen*), but had less activity in the earlier years. His expertise hardly even crossed one

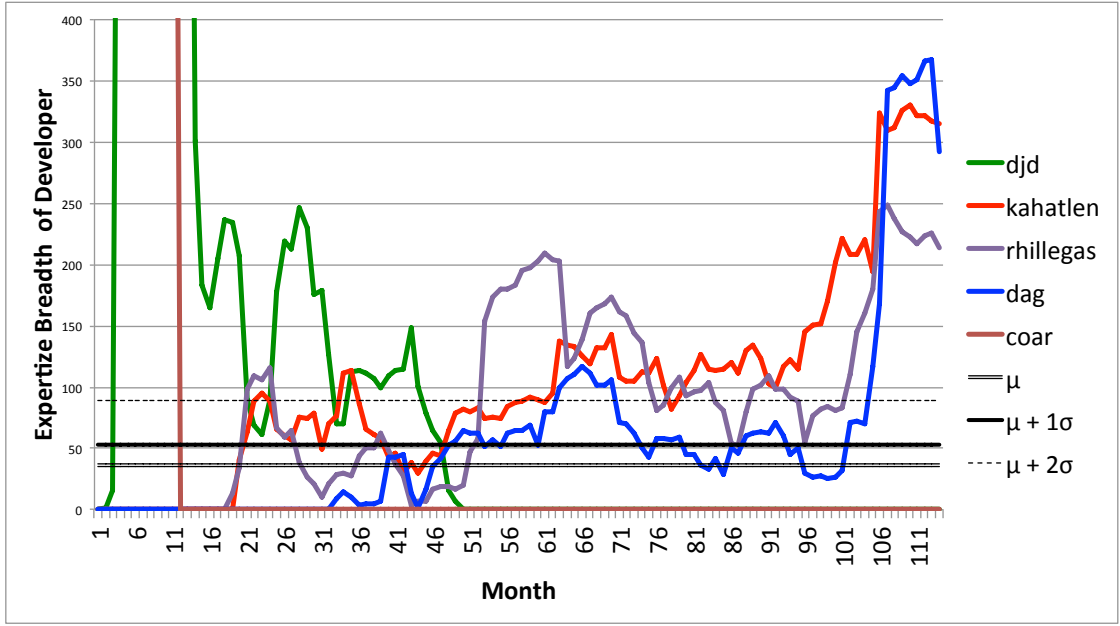


Figure 5.7: Developer breadth expertise for the whole project.

standard deviation above mean (lower solid single line) during the project. Nevertheless, he was an active developer. However, when analyzing over the entire project history, *dag* (see Table 5.12 ED column) is classified as the 8<sup>th</sup> expert in the project.

#### 5.4.4 Performance

Our approach has been designed to enable online exploratory analysis over large-scale software engineering data. It makes efficiency of computation an underlying requirement. In this section, we provide some benchmarks contrasting our approach using Dominoes with an equivalent tool in CPU. This tool was specifically developed to allow comparison of optimal CPU implementations with our GPU implementation.

In order to make a fair comparison, all linear transformations in CPU are made in OpenBLAS <sup>6</sup>, an open source implementation of BLAS (Basic Linear Algebra Subprograms) API with many handcrafted optimizations for specific processor types, including multi-core parallelism features. BLAS are a specified set of low-level subroutines that perform common linear algebra operations that include matrix multiplications. OpenBLAS is able to decompose a BLAS operation into smaller “kernel routines” and is thus able to use all available CPU cores during its processing, thereby making it a fair comparison. We experimented both tools with different matrix sizes, providing evidences regarding scalability. Additionally, in order to calculate the time, we processed the mean after run-

<sup>6</sup>OpenBLAS: <http://www.openblas.net>

ning the experiments 20 times and removing the biggest and smallest times. We use the **Speed up** metric, defined in Equation 5.6 to measure how fast GPU is in relation to CPU.

$$SpeedUp = \frac{Time^{CPU}}{Time^{GPU}} \quad (5.6)$$

The first step, the initial (full) loading of all commits into the database required 15 minutes in order to create the AST, while the parser to identify all changed methods took about 208 minutes. However, we reiterate that this corresponds to a pre-process stage and occurs only once, as future commits are processed in an incremental way.

The dependency calculation, described in Section 5.4.1, produced a matrix  $CCI$  ([commit|class]) of size  $7,578 \times 34,335$ . The generation of  $CCI^{sup}$  and  $CCI^{conf}$  took about 0.2 minutes in GPU. However, performing the same computation using CPU took 413 minutes. This shows that we get a speedup of three orders of magnitude when using GPU - with just the simple calculation that requires one transposition and three multiplication operations. Similarly, when we process  $MC$  (i.e., [method|commit]), with size of  $305,551 \times 7,578$ , it takes about 0.3 minutes in GPU. This calculation was infeasible in a reasonable amount of time when processed on CPU (stopped after waiting for 720 minutes).

When evaluating the time required during expertise evolution across the time, described in Section 5.4.3, we initially measured the total time spent to process the 3D tile introduced in Section 5.3.2 at the coarse level ([developer|file|time]). This tile comprises 114 layers of a  $36 \times 3400$  ([developer|file]) matrix (a total of 13,953,600 elements). Building this tile from Dominoes' database takes 2.38 seconds. On the other hand, composing a 3D tile at fine grain level ([developer|method|time]) takes about 189.96 seconds, comprising of 114 layers of  $36 \times 43,788$  matrices ([developer| method]) each. Tile generation is always done in CPU, as it is a query over the persistent database.

Processing the z-score requires calculating the mean and standard deviation of each layer and then computing the z-score itself. Table 5.13 shows the times taken to calculate EBD. In the first column ("EBD<sup>F</sup>"), we show the time necessary for creating a 3D tile for the entire project when considering files as the unit of analysis. The total time taken to process all layers ("Total" column) is 303.42 versus 19.59 seconds when comparing CPU and GPU, respectively. Therefore, we have a speed up of 15.48 times in relation to CPU. In the second column ("EBD<sup>M</sup>"), we also calculate EBD for the entire project, but this time considering methods as the unit of analysis. The total time taken to process all

layers (“Total” column) is 1,998.31 and 212.01 for CPU and GPU, respectively, achieving a speedup of 9.42 in relation to CPU.

Table 5.13: Processing time (in seconds) for calculating 3D tile for EBD in the project.

	<i><b>EBD<sup>F</sup></b></i>			<i><b>EBD<sup>M</sup></b></i>		
	<i>Mean</i>	<i>ℰ</i>	<i>SD</i>	<i>Mean</i>	<i>ℰ</i>	<i>SD</i>
<b>CPU</b>	2.19		301.23	303.42	424.71	1,573.60
<b>GPU</b>	0.10		19.49	19.59	8.55	203.46
<b>Speed Up</b>	21.90		15.45	15.48	49.67	7.73
						9.42

The results show that a large difference in performance can be achieved when we perform the same analysis in GPU as compared to using CPU, even when the latter employs optimal algorithms provided by OpenBLAS. It is important to note that we used a conventional and affordable GPU card when running the experiments. Replacing this card with a more powerful card, such as nVidia Tesla K40, could easily boost Dominoes performance. The nVidia Tesla K40, the most powerful GPU at the time of writing of this thesis, provides at least 20 times more computational power.

## 5.5 Usability evaluation

In order to evaluate Dominoes as a tool for exploratory analysis, this section presents the approach used for its evaluation. Additionally, the results and insights gathered from this study are also discussed in this section.

In this experiment, we are focused on answering the following research questions:

- How is the influence of the use of derived tiles for getting the right answer?
- Is it important to check the data produced along the exploration?
- What is the influence of relationships representation during exploration?
- What is the behaviour of participants after changing their exploratory path due to mistakes?
- What are the barriers to start using Dominoes?

The remaining of this section presents the methodology used for the evaluation (Section 5.5.1) and the results of our analysis (Section 5.5.2).

### 5.5.1 Methodology

Dominoes evaluation was performed by using a think-aloud method to investigate how participants conduct their thoughts during exploration.

We used a section of the Apache Derby project history and formulated four scenarios (i.e., tasks) to be solved by the participants. This subset of Apache Derby history comprehend a period from January 2013 to January 2014. The selected period produced a total of 602 commits, 1,316 changes in classes, 7,792 changes in methods, and 264 issues. We chose Apache Derby project since it is a stable project, still active since the beginning of its life, and is a long-living project.

During the experiments, we observed participants while they performed actions to solve the proposed problems.

#### 5.5.1.1 Scenarios

Participants were presented to four scenarios to be solved using our tool. All scenarios were aimed at using Dominoes tiles and their connections in order to manage and build repository data relationships. For each scenario, a wide range of exploration paths could be used at different abstraction levels (i.e., file, class, methods, etc.), as well as different ways of composing data. In the following we describe all four scenarios.

Scenario 1: *“Richard is planning on performing a major refactoring over the code he has worked on in the last 3 months. He wants to analyze the commit history of his modifications to identify which developers might be affected by his proposed refactoring. How can he do so?”*. This scenario was made in order to evaluate how is the barrier to starting using Dominoes as well as if its relationship representations were easy to understand by the participants.

Scenario 2: *“Knut has been a core developer in Derby, but lately he has had too many issues to resolve and is not able to fix them quickly enough. Therefore, the Derby manager, Susan, has decided to give Knut a team of developers to work with. Knut would like to form the team with people he has worked with before (in the context of fixing issues in the past 6 months). How can Knut identify the developers who should be on his team?”*. In this scenario, we would like to observe the path taken for each participant, as it is a more open question.

Scenario 3: *“Derby’s project manager, Susan, wants to identify the appropriate de-*

veloper to be assigned to a (new) task that requires significant modifications to the class `java.drda.org.apache.derby.drda.NetworkServerControl`. She wants to do so based on the developer history of the class in the last 4 months. How can Susan identify the developer for this task?”. In this scenario, we aim at evaluating the participant’s ability for composing her answer using different fragments of data by looking at each one individually.

Scenario 4: *“The Derby team has realized that they have not refactored their code base in a while and functionalities have been added in an ad hoc manner, so they would like to refactor their code base. However, they have limited time for doing so, so they want to first identify the classes that are the most brittle - that is, classes that have undergone a lot of changes in the last six months of development. How can the team do so?”*. In this last scenario, we would like to observe if the participant grasps the meaning of each kind of relationship available in Dominoes, as this scenario can only be answered by using specific tiles.

It is important to state that all scenarios are aimed at evaluating the different exploration patterns used by each participant.

#### 5.5.1.2 Participants

In this evaluation, we are interested to know how participants explore repository relationships by using Dominoes. For this reason, we selected novice participants that had never played with repositories before in order to see how they behave when using Dominoes. Moreover, we recruited people with industry experience as well as computer science students, with different academic degree. Additionally, we asked each participant if they have worked with repository analysis before, so we could map how previous knowledge on repository analysis is important for using and understand Dominoes concepts. Figure 5.8 shows this information for each participant.

#### 5.5.1.3 Study Design

A set of steps was established for the evaluation, as presented in Figure 5.9. The first step consists in providing a document to the participant explaining how the experiment is going to be conducted. In the following, they filled in a consentment term for anonymously using and publishing their collected data. In sequence, participants were instructed about the think aloud method and presented to a video tutorial showing how Dominoes works



Participant	Industry Project Experience (Years)	Academic Project Experience (Years)	Experience	
			Academic Degree	Repository Analysis
P1	-	Between 3 and 4	D.Sc.	✗
P2	-	Between 1 and 2	M.Sc.	✗
P3	Between 1 and 2	Above 4	M.Sc.	✗
P4	Between 3 and 4	Between 3 and 4	B.Sc.	✗
P5	Less than 1	Above 4	M.Sc.	✓
P6	Above 4	Between 1 and 2	B.Sc.	✗
P7	Above 4	Between 1 and 2	B.Sc.	✗
P8	Above 4	Above 4	B.Sc.	✓
P9	Above 4	Between 1 and 2	M.Sc.	✓

Figure 5.8: Participants' characteristics.

and its basic operations. Moreover, the participants were reminded about the concepts of matrix multiplication and transposition. All these training steps took around 25 minutes. Finally, we gave the participant five minutes to play with Dominoes, using a different database from the one used in the experiment, before the experiment actually started.

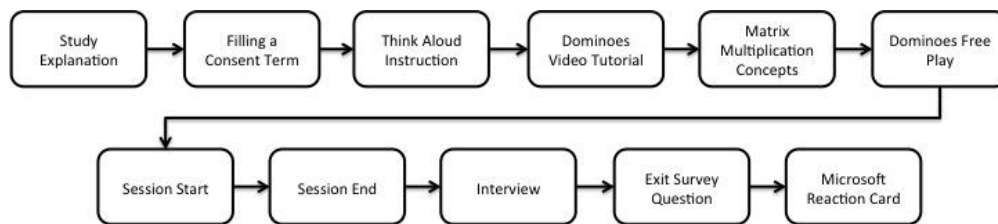


Figure 5.9: Experiment workflow performed for each participant.

After the initial instructions, the participant had a total of 15 minutes for finding the answer to each scenario. Throughout the session, we collected audio of what the participant said, video of the participant while talking aloud, and screen-captured video. In addition, each participant needed to do a printscreen of her solution at the end.

During the session, we took notes about each unusual action performed by the participant, such as finding results in a different way than most of the participants did. After a short break, we conducted an interview with each participant about these notes. We also asked about some mistakes that participants made in order to better understand if the problem was related to Dominoes or due to the participant's lack of knowledge in a specific subject.

Finally, at the end of the interview, each participant received an exit survey form containing the following questions about her experience with Dominoes: (1) *Were Dominoes tiles easy to interact with?* (2) *Were Dominoes derived tiles easy to create and use?*

(3) *Were Dominoes operations easy to use?* (4) *Were Dominoes visualizations useful in answering the questions?*, and (5) *Did Dominoes help you to investigate the Apache Derby project?* These questions were answered based on a likert scale, which ranges from strongly disagree (1) to strongly agree (5). Additionally, the participants used the Microsoft Product Reaction Card <sup>7</sup> to best select their experiences during Dominoes usage.

#### 5.5.1.4 Coding

We analyzed each step performed by the participant by watching the video and transcribing them into concrete actions performed on the tool. In sequence, these actions were coded based on the intention of the participant considering the action itself as well as the context. The codes are shown in Table 5.14, grouped into four main categories: exploration, verification / validation, adjustments, and repetition. These codes were extracted by the author of this thesis and reviewed by other researchers. Our analysis is mainly based on the results of this coding step, however the video was revisited in some other opportunities to double-check the obtained results.

### 5.5.2 Results

In this section we present the behaviour of participants during their sessions while working on the proposed scenarios using Dominoes. As discussed before, their actions were coded to help us to build a map of their intentions, presented in Figure 5.10.

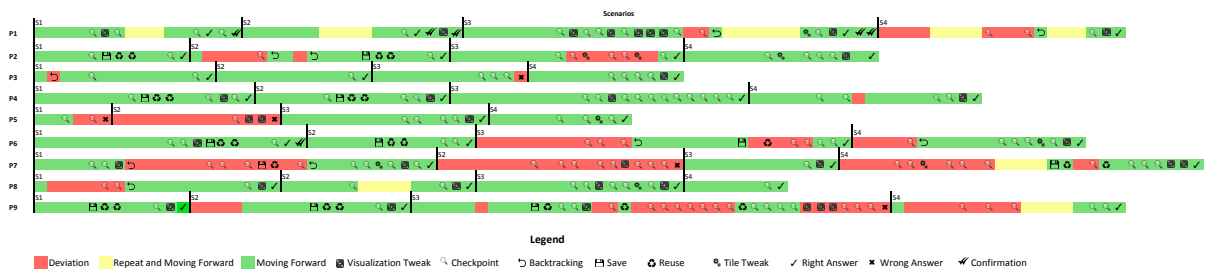


Figure 5.10: Participants' action map for the experiment.

Moreover, Figure 5.11 presents the time (in minutes) spent by each participant during the experiment. For each participant, Figure 5.11 also shows if the participant succeed (a tick) or failed (an x) in the scenario. This information is important for discussing about the actions of the participants during the experiment.

The following subsections answer the aforementioned research questions.

<sup>7</sup>Developed by and ©2002 Microsoft Corporation. All rights reserved

Table 5.14: Coding used for participants' actions.

Category	Name	Description
<b>Exploration</b>	Moving Forward	When the participant performs expected actions towards the solution.
	Deviation	When the participant performs actions that won't lead to answer.
	Backtrack	When the participant gives up a current exploration path.
<b>Verification / Validation</b>	Checkpoint	When the participant verifies if the steps done until now are correct.
	Right Answer	When the participant reaches the right answer.
	Wrong Answer	When the participant reaches a wrong answer.
<b>Adjustments</b>	Confirmation	When the participant checks another visualization to ensure that the answer is correct.
	Vis Tweak	When the participant adjusts some aspects of the visualization.
	Tiles Tweak	When the participant adjusts some tiles.
<b>Repetition</b>	Repeat	When derived tiles are not used, leading to the repetition of the same operations.
	Save	When saving derived tiles.
	Reuse	When using derived tiles that were previously saved.

#### 5.5.2.1 How is the influence of the use of derived tiles for getting the right answer?

One interesting information we found during our analysis is that trying more than one way to analyze relationship helps people to reach the correct answer. This is observed quantitatively by the number of derived tiles used and its relationship with the number of right answers provided. Here it is important to state that each derived tile represents different relationships among artifacts and thus different perspective of analysis. For instance, one could choose to explore the knowledge of a developer by using the files touched, leading to  $[D|F] = [D|C] \times [C|F]$  (coarse grain). On the other hand, another participant can choose to perform this exploration by methods touched by a developer, leading to  $[D|M] = [D|C] \times [C|M]$  (fine grain). In this case, if the participant gets stucked while analysing  $[D|M]$  tile, she can derive  $[D|F]$  and make the analysis over it, as  $[D|M]$  can easily connect with  $[F|M]^T$ .

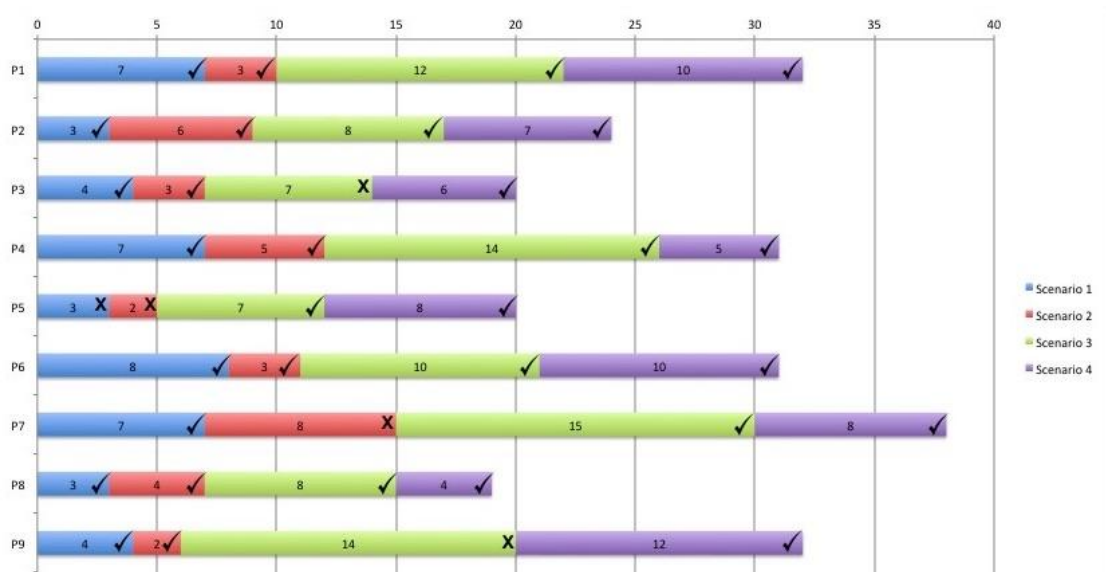


Figure 5.11: Time (in minutes) taken by each participant during the experiment.

Figure 5.12 shows the number of derived and unique tiles and right answer. During the experiment, the minimum number of derived tiles used by a participant was 13, leading to just 2 right answers. On the other hand, the maximum number of derived tiles used by a participant was 29, leading to all scenarios answered correctly.

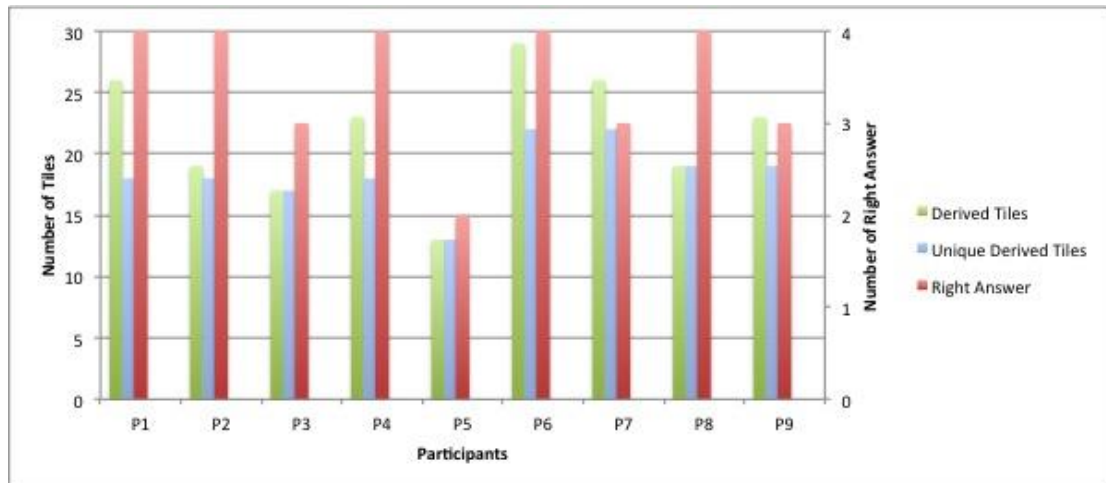


Figure 5.12: Total of derived tiles and unique derived tiles, and right answer.

According to Figure 5.12, it is possible to observe that the number of unique derived tiles closely follows the amount of derived tiles used. Based on this, we can conclude that different exploration was done by the participants while answering the scenarios and normally participants do not get stucked by producing the same relationships repeatedly. A case where the number of derived tiles and unique derived tiles differs is for P1. In fact, when looking at Figure 5.10, P1's experiment is the one with most redundant actions

(repetition, in yellow), showing that previous steps and relationships are often being re-done.

When analyzing P6, it is possible to observe that she is the participant that most used derived tiles in addition to answering rightly all the scenarios. While exploring for giving the answer to a scenario, she tinkered a lot before using the Dominoes tiles (being the fourth participant who spent most time in the whole experiment). From the experiment, we observed that allowing participants to combine tiles helped them to reach more correct answers, as a consequence of visualizing the relationship from different perspectives. P6, for example, said: *“What about if I combine these two pieces [tiles]? Maybe this path will lead to the answer”*.

On the other hand, P5, who had answered only 2 scenarios correctly (S3, S4), was the one with fewest derived tiles. She spent few time to understand each scenario and answering them. Considering the mean time spent by each participant in the scenarios, she just spent about half of this mean time for scenarios 1, 2, and 3. In scenario 4 the time spent was close to the mean. While for scenarios 1 and 2 it is not a big problem (although the participant has answered them wrongly), it is important for scenarios 3 and 4, as they are more complex to answer. Scenarios 1 and 2 require more exploration (Dominoes tiles manipulation) to be answered correctly, being one of the main factor that led P5 to miss them. On the other hand, scenarios 3 and 4 do not require much exploration to be answered (for example, neither of them require using the support operation), which explains why P5 succeed on answering them. According to Figure 5.10, she is the one who did fewer actions to finish the whole experiment and did not explore at all.

However, the presented behaviour (number of derived tiles leading to correct answers) could not be observed for P2, P7, and P8. While for P2 and P8 all questions were answered rightly using few derived tiles, P7 missed one question, even being the second one to use more derived tiles in the experiment.

When revisiting the P2 and P8 video, it was possible to observe that both participants grasp the Dominoes concepts very fast. However, P2 had some difficulties to understand and use the concept of support, requiring some backtracking to fix wrong tiles. Besides this little problem, according to P2 *“Dominoes pieces [tiles] are self explanatory and it is possible to understand relationships easily”*. In addition, P8 had already made repository analysis before and could understand fastly the concepts of relationships in Dominoes (in fact, she spent just 19 minutes for answering all the questions, the lowest time across all the participants. On the other hand, P7 is the participant with the second highest

number of derived tiles and 3 right answers in total. This participant missed answering scenario 2 since expertise based on issue was expected. Instead she computed expertise by looking at the number of file modifications made by each developer. At the end, we asked her why she did not perform the expertise calculation based on issue and she said: *“I did not noticed it [the [issue/commit] tile]”*.

In addition, some paths we did not envision for answering the scenarios happened during this experiment. For instance, while we expected scenario 3 to be answered by looking at a derived tile presenting the total amount of modifications the class suffered by a developer ( $[d|\text{sum}(Cl)]$ ), P1 and P5 answered this scenarios by using a  $[developer|class]$  tile. They used a graph to visualized this tile and played with a threshold until just one edge remained. The same happened to P3, who also used a graph to answer scenario 4.

### 5.5.2.2 Is it important to check the data produced along the exploration?

According to Figure 5.10, it is possible to observe that participants did multiple checkpoints during their explorations. In fact, an average of **16.55** checkpoints per participant have been made for the whole experiment.

When looking at Figure 5.12, P6 is the participant that most used derived tiles. After combining tiles, the participant normally visualized them in order to better understand the problem and take the right direction. In fact, while stucked on scenario 1, P6 opened a derived tile and said: *“That’s it! My logic to answer the question is right and now I know how to proceed”*. On the other hand, during a checkpoint on scenario 3 the same participant said: *“That is not what I want. It’s wrong. I need another dominoes piece [tile]”* and did a backtrack.

In some cases, checkpoints were useful in order to provide information about a wrong path. In this case, participants realized they were in the wrong path and backtracked. In fact, this pattern happened 7 times (4.69% from all checkpoints) in the whole experiment, as can be observed on Figure 5.10. According to Figure 5.13 (zoomed over Figure 5.10 for P2/S2), P2 backtracked twice on scenario 2 after confirming by a checkpoint she was in the wrong direction. In one checkpoint, **the participant recovered to the right direction**. On the other one, the participant recovered to the right direction without performing a checkpoint before. In the first checkpoint, she produced a derived tile that was a diagonal matrix (as said before, P2 had a problem to understand support matrix). After looking at this tile she said: *“It will not help me as it is a diagonal matrix. It does not make sense to be used”*.

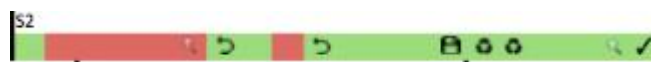


Figure 5.13: Actions performed by P2 on scenario 2.

Additionally, *checkpoint helped participants to keep in the right path* in 94 times (63.09%). For instance, according to Figure 5.10, P4 is the third participant who did most checkpoints. This participant did not deviate any time during the experiment, being able to answer all questions rightly. During her checkpoints, she normally says: *“Ok, I’m getting what I need to answer it [scenario]”*. In addition, checkpoints were normally followed by tiles tweaks for most participants. We observed that this pattern happened 29 times in the whole experiment.

Unfortunately, in some cases we observed that checkpoint was not effective while in deviation. P7 on scenario 2, for example, did a checkpoint for almost every step while deviating in order to see how to reach the correct result. While producing derived tiles, P7 opened a matrix visualization for checking the relationship, sometimes concluding that it was not able to give the correct results. P7 also tried to use different kinds of visualizations for the same derived tile in order to check the data from **different perspectives**. Such an example happened when she was viewing a derived tile using a matrix visualization and then changed to a graph visualization, saying: *“This visualization [graph] is much better to analyze it [relationship]”*.

In order to compare these four cases discussed before, Table 5.15 presents the number of checkpoints performed by the participants in different moments. The first case occurs by doing checkpoints while moving forward and keeping moving forward (area colored in green in Figure 5.10), represented by column 2 on Table 5.15. On the other hand, column 3 represents checkpoints performed while in deviation that did not help to finish the deviation (area colored in red in Figure 5.10). Finally, columns 4 and 5 represent checkpoints performed immediately before switching from deviation to moving forward (switch color from red to green in Figure 5.10) and from moving forward to deviation (switch color from green to red in Figure 5.10), respectively.

Data visualization is important as people like doing progressive evaluation [50]. Dominoes makes it easier through the use of checkpoints. People did it not only when they were deviating, but also when they were making progress (in the right path). According to Table 5.15, this percentage is equivalent to 67.12% (column 2 + column 5). Besides that, considering all participants, in 94% of these cases, checkpoints helped on keeping the participant in the correct path, as only 6% of them deviated just after a checkpoint.

Table 5.15: Checkpoints performed by the participants in different situations.

Part.	Keeping Moving Forward	Keep Deviating	Changing from Deviation to Moving Forward	Changing from Moving Forward to Deviation
P1	11	1	2	1
P2	8	3	1	1
P3	9	0	0	1
P4	20	0	0	1
P5	7	2	0	1
P6	11	3	3	0
P7	10	15	3	0
P8	9	1	1	0
P9	9	11	3	1
Total	94	36	13	6
Perc.	63.09%	24.16%	8.72%	4.03%

Moreover, 32.88% (column 3 + column 4) of the checkpoints were done during a deviation. However, in 26.52% (of the cases they helped the participant to move to the correct path. This result shows a clear tendency of the positive effects of checkpoints: 26.52% of success on correcting a deviation with only 6% of side effect of starting a new deviation.

### 5.5.2.3 What is the influence of relationships representation during exploration?

Although participants used many checkpoints to track their steps towards answering a scenario, some of them (2 participants in 2 scenarios) did a backtracking without doing a checkpoint. For instance, P2 did a backtrack on scenario 2 without looking at the produced data. The same happened on scenario 1 for P3. During our interview session after the experiment, we asked them how they knew they were in the wrong way without looking at the produced data. They told us that the abstraction provided by Dominoes over the data helped them to grasp the meaning of the produced relationships without needing to open it to visualize. They said that by just looking at the Dominoes tile, they could infer that it was the wrong data to answer the question. In fact, this statement was confirmed when we analyzed the video as one of them said: *“It is not the data [that] I need to answer this question”*.



#### 5.5.2.4 What is the behaviour of participants after changing their exploratory path due to mistakes?

Another fact that we could observe from participants while making a backtrack in their sessions is that they normally reduce the time required for using the same number of dominoes tiles after the backtrack (i.e., the density of used tiles by time normally increases). As an example, P1 on scenario 3 placed 0.83 tiles / minute before the first backtrack. After that, she started to place 3.10 tiles / minute. It is represented in Figure 5.14, where the numbers above the line represent the number of tiles placed by time.

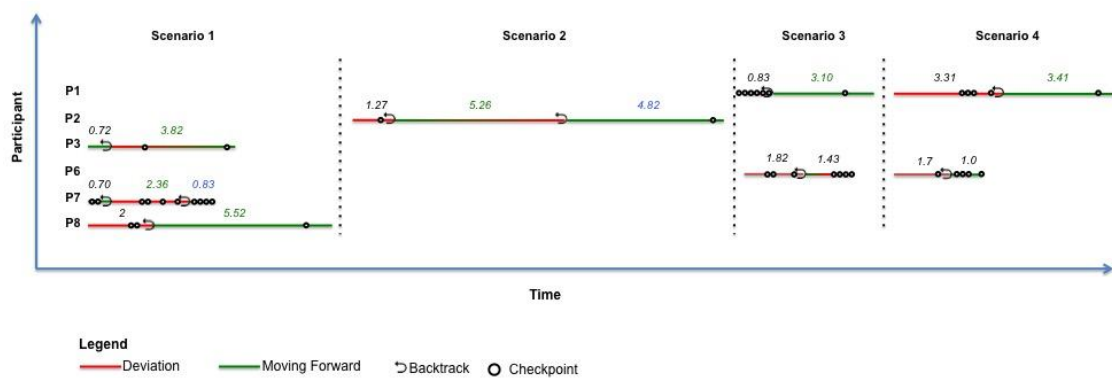


Figure 5.14: Relationship about deviation, moving forward, and backtracking. The numbers over the lines represents tiles / minute.

From Figure 5.14 we can observe that in 6 out of 8 cases, the number of checkpoints performed before and after a backtrack has been reduced. It is important to state that normally a checkpoint requires analyzing the produced data, requiring more time. It shows that after a backtrack, participants already know the produced data, avoiding visualizing it again and, consequently, increasing the number of tiles / minute.

However, the opposite can occur (P6/S3 & S4), i.e., the number of tiles / minute decreased after a backtracking. For these case, it is possible to observe that more checkpoints were done after the backtrack than before it. While analyzing P6 video, we noticed that she was more careful after the first deviation by saying: *“Let me check every step now”*.

Moreover, it is interesting to note that in two cases (P2/S2 and P7/S1) multiple backtracks happened in the same scenario, leading to a reduction of tiles placed / minute after the second backtrack (numbers in blue). By analysing the video of these participants we found that after the second backtracking, they were also more carefully while checking the data for answering the question. In the case of P7, she did multiple checkpoints, explaining the decrease in tiles placed / minute.

From the feedback we received, we can conclude that due to Dominoes performance to analyze and build information, participants did not get afraid of trying out new ways of exploration, and they explored with the tiles pretty fast. According to participant P6: *“Dominoes produces data very fast”*.

#### 5.5.2.5 What are the barriers to start using Dominoes?

When analyzing Figure 5.10, it is possible to see that most of the participants did not have problems at all using Dominoes, as the green color (representing moving forward) is dominant when compared to the red color (representing deviation). Additionally, due to the number of tiles and relationships that Dominoes offers, some low yellow area is observed (meaning repetition).

As told before, at the end of the sessions we asked participants to fill in a form with some closed questions regarding their experience using Dominoes, as presented in Figure 5.15. This form helped us to evaluate Dominoes at the user perspective as well as to understand its weaknesses. Comparing participants that never had contact with repository analysis before (P1, P2, P3, P4, P6, and P7), 2 out of 24 scenarios (6 participants  $\times$  4 scenarios) were not answered correctly (P3/S3 and P7/S2), representing a success rate of 91.66%.

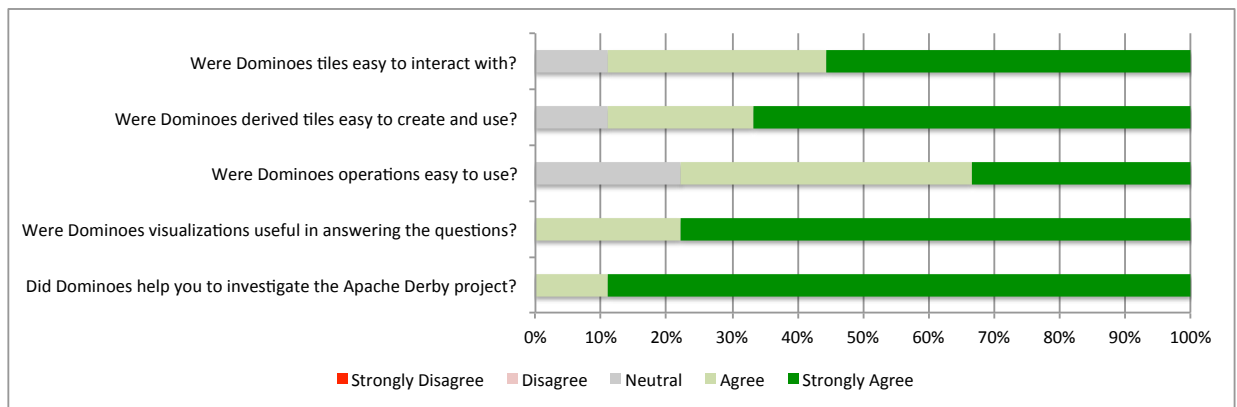


Figure 5.15: Dominoes feedback from the participants.

According to Figure 5.15, 8 out of 9 participants agree that Dominoes tiles were easy to interact with (question 1). In the same way, 8 out of 9 participants agree that creating and using derived tiles was easy (question 2). Moreover, 7 out of 9 participants agree that using Dominoes operations was easy (question 3). Additionally, all participants agree that the visualizations provided by Dominoes are useful for answering the scenarios (question 4). One interesting fact to observe from this figure is that all participants agree (8 out of

9 completely agree) that Dominoes helped them to investigate the Apache Derby project, which was the main objective of the experiment.

Besides that, after filling this form, we asked the participants to choose the words (positive or negative) in the Microsoft Reaction Card that best represented their experience using Dominoes. This selection was used to generate a word cloud presented in Figure 5.16, where the word size reflects the frequency each word was chosen. The most chosen word was “efficient”. As Dominoes uses GPU architecture for accelerating processing, allowing online exploration, efficiency was a key concern in its design and was well recognized by the participants. Additionally, the word “innovative” has been chosen multiple times. In fact, as far as we know, Dominoes is the first approach that uses a ludic metaphor (dominoes game) to allow exploratory analysis over repository. Finally, during the interview at the end of the session, participants told P1 said: *“Congratulations. It [Dominoes] is an interesting tool. When it will be available for use?”*.



Figure 5.16: Word cloud chose by participants from the Microsoft Reaction Card.

### 5.5.3 Discussion

After performing Dominoes evaluation, we could identify some initial evidences that support the following findings:

- Allowing more than one way to explore a problem leads to more correct results during exploration, as each user can select the best grain and relationship that fits to the problem (Section 5.5.2.1).

- Providing the ability to check intermediate data during exploration helps keeping the analysis in the correct path. At the same time, providing these checkpoints helps users to recover from a wrong to the right path (Section 5.5.2.2).
- The use of Dominoes' game pieces for representing the data and relationship creation helps users to understand the meaning of the data and relationships (Section 5.5.2.3).
- People normally are faster after the first backtracking as they already know how to produce data and do not need to visualize it frequently (Section 5.5.2.4).
- As the first contact with Dominoes, users succeed on 31 out of 36 scenarios (Section 5.5.2.5).
- Dominoes seems to be useful, fast, and efficient according to the participants. At the same time, it was easy to interact and compose new relationships (Section 5.5.2.5).

However, our study was also able to identify some deficiencies in Dominoes. For example, P1, P2, P4, P6, and P7 tried to answer scenario 3 by looking at a matrix visualization and by manually searching for the answer. Due to the fact that this matrix is big (about  $8 \times 300$ ), it became difficult for finding the right cell. In this case, after spending some time in this visualization, they tried the bar chart visualization, with more tweaks, such as search and ordering. For instance, while looking at the data using the matrix visualization, P1 said: *"It is important to have an option to search on this visualization"*. Additionally, P2 said regarding the bar chart: *"It would be interesting to put the number on each bar in the bar chart. Depending on the numbers of bars, it is difficult to see the numbers"*.

Moreover, although interacting with the tiles was intuitive, we could observe some trouble when moving pieces to the editor. People wanted to drag the tiles to the editor, however the *hand* icon only means that tiles can be reorganized in the library, not dragged to the editor (this is done with a double-click). Participants quickly understood the interaction paradigm, but it is something to change in the future.

## 5.6 Threats to validity

The divergence of expertise from expertise breadth that we observed is largely based on the Apache Derby project. We selected Apache Derby because it has a large and active contributor base. It is possible that other projects might not have as much as deviation

in expertise breadth. We need further studies of other projects to observe whether such deviation between expertise and expertise breadth holds true. A central construct in our analysis is that if someone edits a method, we assume that the developer has a certain degree of knowledge of that method. However, since other approaches also use number of edits as a proxy for knowledge, this is not a major threat. Further, our analysis filters out non-source code changes to the method (or file).

Another threat in our study is that we use z-scores to identify expertise, which assumes a normal distribution. Project data might not be normal. However, since we use z-scores to identify developers who have made edits more than the average number of times, non-normality of our data should not be a problem. In the worst case, the z-score provides us a ranking (even if some developers have negative scores) with a perception of the distance among data points in the sample.

Additionally, each method inside a class might not have the same importance. For instance, get/set methods have a low complexity to implement or maintain, when compared to other functional methods (e.g., *executeConstantAction*). In this case, it is desirable to use a weighting mechanism for methods according to their complexity, which can then help in identifying experts not only on the breadth of their knowledge, but also on the complexity of the changes that they have made.

Regarding to usability study, we received positive feedback from the participants that used Dominoes. However, this study was executed with only 9 participants, producing a total of 36 scenarios. Although almost all their thoughts and actions were carefully analyzed, we cannot generalize the results due to the low number of participants and their representativeness. In this case, we cannot assume that Dominoes will be as useful as discussed here for other users or in other scenarios. Additionally, the participants are familiar to software engineering field, facilitating their analysis and comprehension of the tool.

## 5.7 Related work

We could identify a broad range of related work focused on data exploration for knowledge. Some related work contributes by identifying dependencies [22], providing tools for exploratory analysis [93, 13, 44, 35], expertise identification [46, 79, 4, 65, 91, 96], proposing infrastructure for repository analysis [12, 82, 45], and infrastructure focused on speeding up data analysis [36, 41].

Determining dependencies can be done by a number of approaches that focus on identifying structural dependencies (through syntactic analysis) or logical dependencies (through change history) amongst artifacts. Cataldo et al. [22] stands out as they use matrices to process dependencies among developers based on dependencies among artifacts. In their approach, both structural dependencies and logical dependencies become Task Dependency ( $T_D$ ) matrices, and change requests, associating developers to artifacts, becomes Task Assignment ( $T_A$ ) matrix. These matrices are used in an equation that indicates coordination requirements  $T_A \times T_D \times T_A^T$ . Our approach generalizes this idea by allowing different kinds of exploration over matrices. Finally, our identification of relationships is innovative, as it allows combining support, confidence, and lift, to compose the dependency matrix depending on the required analysis.

Gîrba et al. [46] defined an “Ownership Map” visualization to understand when and how different developers interacted and in which parts of the code, as well as define who a file belonged to and for how long. In their approach, lines of code added/removed by each developer are counted to determine ownership. A developer who owns (made the latest edit) the most number of lines in the file is considered the owner of the file. It does not take into account the syntactic structures of the file (such as classes or methods) and is influenced by non-functional changes like comments. Our approach, in contrast, can detect changes at different granularities as well as ignores non-functional changes.

Exploratory Data Analysis tools provide either predefined questions or are very limited to derive information that was not conceived beforehand. In the case of Tesseract [93], for example, the available relationships are preprocessed and the matrices are fixed at a coarse grain (file-file, file-developer, file-bug, bug-developer). CodeBook [12] follows a similar approach and creates a network that connects developers and artifacts by mining version control change logs, emails, and other artifacts in a software repository. The underlying graph can then be used by applications to answer different analysis questions (e.g., WhoseIsThat [13] identifies artifacts edited by a developer). Additionally, exploration using CodeBook is constrained to search nodes and their connections. Our approach uses matrices to define software relationships and allows for unconstrained data exploration and manipulation. Moreover, we perform incremental update of data, which is not the case for CodeBook.

Information Fragment [44] allows a user to compose information from tasks, change sets, and teams to explore the relationships between these entities. Unfortunately, it operates at a predefined granularity level, while navigating from fine-grain to coarse-grain

and vice-versa is essential for exploratory analysis. Also, it needs to restrict the data that can be analyzed, because performing interactive data analytics of software archives through visual explorations of relationships among project elements is infeasible at the scale of operation that is needed.

Recommender Systems are approaches designed to help decision-making. McDonald and Ackerman [79] introduced Expertise Recommender (ER), which is based on two heuristics for recommending developers for specific tasks: tech support and change history. The tech support heuristic uses an issue database to search for similar situations and recommends the people involved in previous situations. The change history heuristic states that the last person that changed an artifact is a good candidate for changing it again. Unfortunately, the latter heuristic places a high weight on the most recent changes and ignores the past, which might affect the quality of the recommendations. Our approach uses the entire history of a repository, but segregates this history into timeframes that allow the perception of how expertise fluctuates over time.

Anvik et al. [4] proposed an approach to recommend developers for a specific issue by using machine learning techniques exclusively over an issue database. This way, the provided information does not take into account artifacts' modifications in order to suggest a developer who is the most appropriate to change specific artifacts. Our focus is different, as we identify expertise over artifacts (e.g., files, project).

Kagdi et al. [65] propose a system for assisting in the tasks of allocating developers for changing a given file. It considers three metrics to compose a ranked list of recommended developers: contribution, activity, and recency of changes. The contribution metric indicates the number of commits each developer has made over a file. The activity metric indicates the number of days the developer has committed at least once in the project. The recency metric indicates the date of the last commit of each developer. This approach works only at coarse grain (i.e., files) and is not designed to support online exploratory analysis. Further than a recommendation system, Dominoes provides a generic and flexible platform for exploratory analysis of project elements at any granularity level, which is compatible with multiple data types and relationships. Its interactive capabilities are mainly possible due to the adoption of the massively parallel architecture of the GPU.

Posnett et al. [91] consider both artifact and developers perspectives in order to extract focus and ownership for computing a unified score: DAF (developer attention focus), which measures how focused is a developer during his task (i.e., their work is spread among many artifacts or is more focused). In contrast, EBD measures the breadth

of expertise of a developer in specific artifacts (or the project).

Schuler and Zimmermann [96] propose an approach for measuring expertise by the frequency in which a developer uses a method. While this information can help for identifying a person who knows how to use a method, it might not help in identifying the developer who knows that method the best, that is, the person who is the most appropriate to edit that method. Instead, our approach uses modifications inside a method for proposing experts in a file.

Additionally, there are works that propose infrastructures to facilitate the automated expertise identification through repository exploration. For example, Minto and Murphy [82] introduce the Emergent Expertise Locator (EEL). EEL is based on the framework by Cataldo et al. [22] for matrix manipulation, thus requiring massive linear operations to be performed depending on the size of the project. To avoid this problem, EEL imposes a constraint over matrix size, allowing only matrices up to  $1,000 \times 1,000$  elements to be used. Besides that, EEL uses coarse-granularity (i.e., files) to recommend experts. However, the problem of personnel allocation becomes harder for large projects, with much more than 1,000 files. Moreover, assuming files as atomic units may lead to inadequate recommendations, as changes in very specific parts of the file or broader changes in multiple parts are considered equivalent. Our proposed approach, on the other hand, works at a fine grain (i.e., methods) to differentiate specific changes from broader changes. This leads to large matrices to be processed, which can be performed interactively because of the underlying GPU architecture of Dominoes.

Evolizer, by Gall et al. [45], is a tool for mining software archives at fine grain in order to compare source code changes. From these analyses, recommendations such as change type patterns and consistency of changes can also be made. It analyzes AST level changes to identify different types of changes and modification patterns. Dominoes allows for an open set of questions that can be answered based on how the relationships are composed by the user during exploration. Also, our architecture is modeled in a way that any repository can be plugged into the system, avoiding the necessity to build a new tool. Tempe [35] is a tool for data science environment exploration. It is based on a query language and allows for living programming, providing continuous visualizations as well as data streaming. However, it is important to state that data used in Tempe for analysis has to be processed before its usage. On the other hand, Dominoes provides the user with processed data that can be used for manipulation, analysis, and visualization.

Finally, there are works that aims at speeding up repository analysis. Boa [36] provides



an infrastructure for analyzing large scale software repositories on a cluster. Although its performance may be comparable to Dominoes, setting up a CPU cluster is not a trivial task, making it more applicable for researchers in the university than for developers in the industry. On the other hand, when an infrastructure is already built to be used by the research community, other constraints can be applied such as confidentiality over the data, which can only be used inside the organization. Jean-Rémy et al. [41] developed the Harmony platform, a unified model that extracts and analyzes data at a coarse grain from different version control systems. After extracting the data to a database, the user is responsible for dealing with pieces of data to extract the desired information. These tools require the user to write a functional script to define what the platform needs to process per analysis type. This can be a constraint, since end users will need to program their analysis and write a script for each of their queries. Dominoes has a friendly user interface that allows even non-programmers to start exploring a repository.

## 5.8 Final considerations

Dominoes is an exploratory data analysis approach that allows users to select information about different project elements and their interrelationships from a repository. Relationships are represented by matrices, defined as basic building tiles and derived building tiles. Both kinds of building tiles can be combined iteratively to reveal deeper and complex relationships. Through such explorations, relationships that have not been computed or published before can be discovered. Dominoes makes possible a new realm of exploratory analysis that can be made at the users own computer, without relying on difficult commands and a complex infrastructure. As all operations are performed in parallel over GPU, exploratory analysis can occur seamlessly at interactive rates, even when computing relationships in fine-grained data. The current version of Dominoes tool extracts data from a Git repository and operates over the matrices by using GPU kernels implemented in CUDA.

The Dominoes architecture was intentionally designed to easily accommodate the definition of new basic building tiles, such as relationships mined from communication channels (e.g., email, chat, discussion forums). The same extensibility feature also applies for operations. Besides the basic matrix operations, such as multiplication and transposition, specialized operations can also be easily created and plugged into Dominoes, as showed in Section 5.2.3 for support, confidence, and lift. This makes Dominoes a contribution to the scientific community, as empirical studies can be reproduced over different corpora in

order to validate an investigation. This has the potential of alleviating the pain of setting up an environment for each trial of an investigation.

Using Dominoes, we contrasted the use of support alone and the use of support and confidence to distinguish the dependence directions. In the case of Apache Derby, employing confidence leads to a more accurate analysis for finding dependencies among artifacts. Moreover, using confidence for thresholding a relationship is more natural for the user, as it represents a normalized value.

Additionally, we demonstrate that performing exploratory data analysis in software repositories has computational challenges. Here we show how our approach can analyze fine-grained data, such as edits to classes or methods to identify expertise in a project. Our results show that when we consider expertise by only recognizing edits at the file-level, we get a 28% deviation as compared to when we analyze expertise based on the breadth of developers' knowledge. Moreover, since we calculate changes at a fine-grained level, we are able to filter out those changes that do not affect the method body, thereby, being more precise in expertise identification.

We also presented the concept of tridimensional matrices of relationships across software project elements over time. These matrices allow temporal analysis of relationships. When identifying expertise in Apache Derby, we found that temporal analysis shows the flow in developer expertise for a single file, as well as for the whole project. It also shows when a developer stopped being active and “handed over” the expertise to another. Had we considered the entire history of the Derby project, the analysis would incorrectly recommend a developer who was no longer active in the project.

Finally, the Dominoes' usability had been evaluated. In this evaluation, participants explored a project that they never heard about before, producing new derived tiles, different relationships, and new perspectives for visualizing these relationships. In such project, it was observed a rate of 86.11% of right answers, which is a positive score for a new tool. Based on this result, a natural extension would be to release Dominoes to the community in order to get feedbacks about how people behave using Dominoes for exploring their own projects for finding different kinds of relationships.

# Chapter 6

## Conclusion

### 6.1 Contributions

In this thesis, we present two negative consequences on VCS due to limitations related the processing of large amount of data: (1) the lack of *diff*, *patch*, and *merge* support for binary artifacts (more specifically for images and videos), and (2) inability to perform on-line exploratory repository analysis over the project history.

Binary artifacts are typically treated as opaque data by VCS. In this case modifications performed over these binary artifacts are difficult to comprehend in addition to make use of more memory due to the lack of a *delta*. Moreover, a VCS repository can be used for extracting information such as relationship dependencies and expertise and its fluctuation over time. However, depending on the repository size and its lifetime, some constraints to extract information are imposed such as the grain of processing and history limitation, which normally affect the accuracy of extracted knowledge.

As presented in Chapter 1, the goal of this thesis consists on conceiving, implementing, and evaluating GPU algorithms and data structures for solving the aforementioned problems, offering VCS support for: (1) *diff*, *patch*, and *merge* operations over image and video artifacts and (2) fast processing of large repositories for knowledge extraction. To reach this objective, a great amount of data needs to be processed, whether for allowing *diff*, *patch*, and *merge* support to binary artifacts or extracting knowledge. In this case, our contribution can be dismembered on processing binary artifacts, as discussed in Section 6.1.1, and extracting knowledge over repositories, presented in Section 6.1.2.

### 6.1.1 *Diff, patch, and merge over image and video*

Processing binary artifacts on VCS requires specialized *diff*, *patch*, and *merge* operations. In this thesis, an approach for dealing specifically with image and video artifacts was proposed. This approach makes collaborative work possible for image and video artifacts, enabling merging, and indicating, through observation, any physical conflicts that may exist. Additionally, besides the visual information about the change provided by the *delta*, it significantly reduces the space required for storage when compared to saving the whole artifact. In this way, our *delta* reduced the disk space requirement up to 99.57 % for video, and even decreased the repository size for images when a consecutive blur filter is applied.

Due to the fact that processing these kinds of artifacts requires a significant effort and may jeopardizing the user productivity, our approach uses GPU architecture. Our GPU alternative can boost VCS common operations over images, reaching a speed boost almost  $55\times$  higher in some cases, when compared to a CPU-bound architecture. When analyzing the time for processing a video artifact, the boost can be up to  $2.2 \times$  higher for some operations. Our proposal is designed and implemented using an architecture that allows its easy inclusion in commercial, existing VCS.

We can summarize the benefits of our contributions on *diff*, *patch*, and *merge* over images and videos as:

- Reduction by approximately 10 times of the repository size when using *IMUFF*;
- Reduction by approximately 14 times of the repository size when using *IMUFF* image processing techniques;
- A speedup performance up to  $55 \times$  in GPU in relation to CPU in *IMUFF*;
- A file size reduction about 99.57% when comparing the size of the *delta* to the size of the whole video artifact by *VIMUFF*;
- A speedup performance up to  $2.55 \times$  in GPU in relation to CPU in *VIMUFF*;

### 6.1.2 Repository analysis

In this thesis we presented Dominoes, an exploratory data analysis approach that allows users to select information about different project elements and their interrelationships

from a repository. Additionally, the concept of tridimensional matrices of relationships across software project elements over time was employed. These matrices allow temporal analysis of relationships.

Due to our GPU approach, all these explorations over repositories can be performed fast when compared to the CPU. Our approach is able to overcome CPU in about three orders of magnitude during repository processing.

We can summarize the contributions on repository analysis as:

- Design and implementation of a tool (Dominoes) for on-line repository analysis;
- Usage of Dominoes to analyze artifact dependencies on Derby project;
- Usage of Dominoes to analyze developer expertize on Derby project;
- Usage of Dominoes to analyze developer expertize fluctuation over time on Derby project;
- Benchmark over Dominoes showing that the use of GPU can reach speedups up to  $49.67 \times$  in relation to CPU, allowing interactive features in repository exploration; and
- Usability evaluation of Dominoes, showing a success rate of 86.11% while participants explored a repository using Dominoes for the first time.

It is worth it to mention that Dominoes library is being used by other projects in the group, that aim at identifying the best developer to merge branches [34].

## 6.2 Limitations

*IMUFF* can deal with many different types of images. However, currently it only works with image artifacts that are of the same size, generating results that are also of the same size. To bypass this constraint, a modification of the data structure permitting it to deal with images of different sizes is required. In such a case, only portions of the images that are actually being processed need to be stored. It is important to state that this does not affect the size in terms of disk space, due to the adoption of compression prior to storing the generated *deltas*. This limitation is due the fact that our solution is working a pixel-by-pixel matching.

Regarding *VIMUFF*, our approach for video artifact manipulation, some aspects could be improved. One of them is related to frame storage during extraction. Currently, all frames extracted from a video are maintained in memory. This could impose a barrier over the size of the videos that can be processed by *VIMUFF*. In order to solve this, each video frame can be maintained in disk, being indexed by the hash of its content, allowing retrieval of a frame in  $O(1)$  time. Additionally, the MD5 based hash takes a considerable amount of time to be generated according to our results. Replacing it by a parallel version of SHA-1 hash [42] can improve *VIMUFF* during its processing.

Besides that, some steps could be further parallelized, such as the *DCT hash* generation. Currently, we compute one frame at a time in GPU, which degrades the performance due to memory latency. One solution for this is computing in batch or using the CUDA stream mechanism [88]. We believe that this modification would greatly improve the performance of this step.

Additionally, one of the biggest problems when using *VIMUFF* with compressed video format is regarding pixel alignment. In this case, two frames that look like the same have pixels with different colors due to quality loose during the previous compression. Applying *VIMUFF* over them generates a high number of modified frames, which highly increases the *delta* size, sometimes at values that is five times bigger than summing up the two original videos. For solving this problem, a filter can be applied over the frame to minimize the noise and avoiding such high number of not aligned pixels.

Dominoes uses a matrix approach for representing relationship among artifacts, thus providing a set of operations that make sense in terms of matrix and data analysis. In this case, the existence of a relationship among artifacts is represented as one, or zero otherwise. This representation still makes sense after a matrix manipulation, as multiplication by zero produces zero, indicating no relationship. However, some kinds of relationships cannot be represented this way, such as a file deletion. For instance, using another number to represents a file deletion in [commit|file] tile would lead to incorrect results after its multiplication. For solving this problem, a non-binary representation must be used and, keeping valid at the same time the matrix operation.

Moreover, regarding to Dominoes' visualization, it would be interesting to allow more options for it, such as search and sort. This was one of the most problems participants faced while doing their analysis.

## 6.3 Future work

*IMUFF* is capable of producing understandable *delta* for presenting the difference between two images. However, when global transformations are applied over the entire image, it leads to a *delta* without black pixels, which may be difficult for users to understand the difference between two images. In order to minimize this problem, we try to detect such global transformations (i.e., geometric transformations that are applied to the image as a whole). If this global transformation cannot be retrieved, another approach should be used for avoiding presenting to the user an incomprehensible *delta*. One possible solution is performing object detection over an image and tracking its modification such as moving its position or reducing its size. Being able to detect such kind of transformation would greatly benefit the user by providing a more precise identification of changes. This modification would impact the data structure, requiring the storage of such specific object transformation. Unfortunately, algorithms for object detection and tracking are related to computer vision, which are normally complex, requiring a considerable amount of time. However, some efficient algorithms to detect objects and transformations do exist and may be implemented using GPU for boosting their processing.

Additionally, our presented *delta* shows the result of a XOR operator, where the color is different from the one observed in the image. In order to improve the comprehension of the difference, a mask could be produced by using the pixels' position presented in the XOR and applied over the original image, thus exhibiting the real color of the pixels instead of the XOR result.

Finally, the *IMUFF* could be applied for different situations other than VCS. For example, it could be adapted for detecting common areas in the image or identifying duplicated regions within an image.

Currently *VIMUFF* cannot deal with sound stream presented in video artifacts, just operating over the video stream. For this reason, a natural future work would be developing *diff*, *patch*, and *merge* operations for sound artifacts and integrating them to *VIMUFF*, allowing it to process all the streams presented in a video. This would allow a complete solution for versioning video artifacts.

*VIMUFF* works well for small videos like the ones used for evaluation. However, when the video presents a long duration, such as films, it may take considerably more time. One possible solution for this problem could be organizing frames into chunks. Instead of calculating the hash for each frame, the hash would be calculated by chunks.

The number of chunks can be generated by applying a recursive subdivision heuristic in order to find the longest common chunks. In this case, a video can be processed by using a dynamic granularity, instead of working over all frames of the video at once.

Additionally, many steps are still being executed in CPU, leading to a small speed up when compared to *IMUFF*. The MD5 hash generation is the slowest step executed by *VIMUFF*, actually processed in CPU. Using a parallel algorithm for generating such hash code in GPU would heavily improve the whole time required for processing a video in *VIMUFF*.

Another possibility for *VIMUFF* is performing an internal *diff*, which represents a *diff* inside the same video. For this situation, the user could lock a frame, or a chunk, and process the difference between it and other chunks or frames in order to detect significant changes between the locked frame or chunk and the others. One application for such internal *diff* is checking if a security camera has recorded a movement without having to look at the whole video.

Finally, some video codecs use the concept of *keyframes*, which is a frame stored as a whole followed by a certain number of frames stored as a difference to the *keyframe*. When processing such videos, *IMUFF* could use these *keyframes* and all the following deltas to speed up the processing without the need to uncompressing the videos.

Regarding both image and video artifacts, an interesting work would be creating a multimedia repository specialized in such artifacts. In such case, all approaches presented so far for image and video artifacts could be used for multimedia documents with reasoning support.

Dominoes allows repository exploration through the manipulation of its tiles in order to produce relationships. During our study, we make use of some of them to analyze artifact dependency and expertise and its fluctuation in time. However, more analysis could be made with the others tiles that are available in Dominoes. For instance, we did not make any kind of analysis involving the [commit|issue] title. By using this tile, it should be possible to extract information about buggy artifacts and its evolution with time. Additionally, the manipulation of this tile could be used to check how the team could be organized in order to minimize the number of bugs. On the other hand, bugs added by a developer could be also related with the expertise of this developer. By correlating bugs with expertise it would be possible to see through time if increase in expertise decreases or not the number of bugs added by a developer.



In addition, Dominoes is ready for importing new data from another sources, such as email communication. Expanding the sources of data available in Dominoes can further increase the types of analysis. Using a relationship about users and her communication can unveil certain patterns about team organization. Besides that, it can be used with [commit|issue] tile to analyze whether or not communication decrease bugs.

Finally, Dominoes approach can be applied to other domains by letting users to load and play with their own matrices representing other kinds of relationships. In this case, all operations and infrastructure provided by Dominoes would be available for these matrices. Following this idea and based on the fact that we also propose fine-grained processing both image and video artifacts, Dominoes could be adapted for doing repository exploration over such artifacts instead of just source code. When considering a video, for instance, we could still work at coarse-grain, by considering the video as a whole, or at fine-grain, by considering the frames of the video. Even further, some exploration could be done at pixel level in order to detect information such as the most changed area (which may represents an object at high level) and the most conflicted areas during a merge.

# References

- [1] Ieee standard glossary of software engineering terminology. *IEEE Std 610.12-1990* (1990), 1.
- [2] AKENINE-MÖLLER, T.; HAINES, E.; HOFFMAN, N. *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA, 2008.
- [3] AMMAR, S. R. L. F. M. A. R. A. *Multicore Computing: Algorithms, Architectures, and Applications*. CRC Press, 2013.
- [4] ANVIK, J.; HIEW, L.; MURPHY, G. C. Who should fix this bug? In *International Conference on Software Engineering* (New York, NY, USA, 2006), ICSE '06, ACM, pp. 361–370.
- [5] ANVIK, J.; MURPHY, G. C. Reducing the effort of bug report triage: Recommenders for development-oriented decisions. *ACM Trans. Softw. Eng. Methodol.* 20, 3 (Aug. 2011), 10:1–10:35.
- [6] ARMAN, F.; HSU, A.; YEE CHIU, M. Feature management for large video databases. In *SPIE Storage and Retrieval for Image and Video Databases* (1993), pp. 2–12.
- [7] ASKLUND, U.; BENDIX, L. A software configuration management course. In *ICSE Workshops on SCM 2001, and SCM 2003 conference on Software configuration management* (Berlin, Heidelberg, 2003), SCM'01/SCM'03, Springer-Verlag, pp. 245–258.
- [8] AUSTERBERRY, D. *Digital asset management*. Elsevier, 2004.
- [9] BABICH, W. A. *Software Configuration Management: Coordination for Team Productivity*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.
- [10] BARRETO, A. S. *UMA ABORDAGEM PARA DEFINIÇÃO DE PROCESSOS BASEADA EM REUTILIZAÇÃO VISANDO À ALTA MATURIDADE EM PROCESSOS*. Tese de Doutorado, COPPE/UFRJ, 2011.
- [11] BAY, H.; ESS, A.; TUYTELAARS, T.; VAN GOOL, L. Speeded-up robust features (surf). *Comput. Vis. Image Underst.* 110, 3 (June 2008), 346–359.
- [12] BEGEL, A.; KHOO, Y. P.; ZIMMERMANN, T. Codebook: Discovering and exploiting relationships in software repositories. In *ACM/IEEE International Conference on Software Engineering - Volume 1* (New York, NY, USA, 2010), ICSE '10, ACM, pp. 125–134.

- [13] BEGEL, A.; PHANG, K. Y.; ZIMMERMANN, T. Whoselsthat: Finding software engineers with codebook. In *ACM SIGSOFT International Symposium on Foundations of Software Engineering* (New York, NY, USA, 2010), FSE '10, ACM, pp. 381–382.
- [14] BIOLCHINI, J.; MIAN, P. G. Systematic Review in Software Engineering. Tech. Rep. RT-ES 679/05, COPPE/UFRJ, Rio de Janeiro,RJ,Brasil, 2005.
- [15] BOLINGER, D.; BRONSON, T. *Applying RCS and SCCS*, 1 ed. O'Reilly & Associates, Inc, 1995.
- [16] BONANNI, L.; XIAO, X.; HOCKENBERRY, M.; SUBRAMANI, P.; ISHII, H.; SERACINI, M.; SCHULZE, J. Wetpaint: scraping through multi-layered images. In *International conference on Human factors in computing systems* (New York, NY, USA, 2009), CHI '09, ACM, pp. 571–574.
- [17] BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).
- [18] BRANDAO, A.; BRANDAO, L.; NASCIMENTO, G.; MOREIRA, B.; VASCONCELOS, C. N.; CLUA, E. Jecripe: stimulating cognitive abilities of children with down syndrome in pre-scholar age using a game approach. In *International Conference on Advances in Computer Entertainment Technology* (New York, NY, USA, 2010), ACE '10, ACM, pp. 15–18.
- [19] BUFFENBARGER, J. Syntactic software merging. In *Selected Papers from the ICSE SCM-4 and SCM-5 Workshops, on Software Configuration Management* (London, UK, UK, 1995), Springer-Verlag, pp. 153–172.
- [20] BUSCH, D. D. *David Busch's DSLR Movie Shooting Compact Field Guide*. Cengage Learning PTR, Natick, MA, USA, 2012.
- [21] CARROLL, S. R.; CARROLL, D. J. *Statistics made simple for school leaders data-driven decision making*. Scarecrow Press, 2002.
- [22] CATALDO, M.; HERBSLEB, J. D.; CARLEY, K. M. Socio-technical congruence: A framework for assessing the impact of technical and work dependencies on software development productivity. In *ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (New York, NY, USA, 2008), ESEM '08, ACM, pp. 2–11.
- [23] CATALDO, M.; WAGSTROM, P. A.; HERBSLEB, J. D.; CARLEY, K. M. Identification of coordination requirements: Implications for the design of collaboration and awareness tools. In *Conference on Computer Supported Cooperative Work* (New York, NY, USA, 2006), CSCW '06, ACM, pp. 353–362.
- [24] CHACON, S. Git-media, 2009. [Online; Accessed in 02-sep-2014].
- [25] CHE, S.; BOYER, M.; MENG, J.; TARJAN, D.; SHEAFFER, J. W.; SKADRON, K. A performance study of general-purpose applications on graphics processors using cuda. *J. Parallel Distrib. Comput.* 68 (October 2008), 1370–1380.
- [26] CHEN, H.-T.; WEI, L.-Y.; CHANG, C.-F. Nonlinear revision control for images. *ACM Trans. Graph.* 30 (August 2011), 105:1–105:10.

- [27] CONRADI, R.; WESTFECHTEL, B. Version models for software configuration management. *ACM Comput. Surv.* 30 (June 1998), 232–282.
- [28] CORPORATION, N. Nvidia cuda programming guide, 2011.
- [29] DA SILVA, J.; CLUA, E.; MURTA, L.; SARMA, A. Niche vs. breadth: Calculating expertise over time through a fine-grained analysis. In *International Conference on Software Analysis, Evolution and Reengineering (SANER)* (March 2015), pp. 409–418.
- [30] DA SILVA JR., J. R.; CLUA, E.; MURTA, L.; SARMA, A. Multi-perspective exploratory analysis of software development data. *International Journal of Software Engineering and Knowledge Engineering* 25, 1 (2015), 51–68.
- [31] DA SILVA JUNIOR, J. R.; CLUA, E.; MURTA, L. Efficient image-aware version control systems using gpu. *Software: Practice and Experience* (2015), n/a–n/a.
- [32] DA SILVA JUNIOR, J. R.; PACHECO, T.; CLUA, E.; MURTA, L. A gpu-based architecture for parallel image-aware version control. *European Conference on Software Maintenance and Reengineering 0* (2012), 191–200.
- [33] DART, S. Concepts in configuration management systems. In *International workshop on Software configuration management* (New York, NY, USA, May 1991), ACM, pp. 1–18.
- [34] DE SOUZA COSTA, C. Método de alocação de participantes para o merge de ramos. unpublished thesis, 2014.
- [35] DELINE, R.; FISHER, D.; CHANDRAMOULI, B.; GOLDSTEIN, J.; BARNETT, M.; TERWILLIGER, J.; WERNISING, J. Tempe: Live scripting for live data. In *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (October 2015), pp. 137–141.
- [36] DYER, R.; NGUYEN, H. A.; RAJAN, H.; NGUYEN, T. N. Boa: A language and infrastructure for analyzing ultra-large-scale software repositories. In *International Conference on Software Engineering* (Piscataway, NJ, USA, 2013), ICSE '13, IEEE Press, pp. 422–431.
- [37] ESTUBLIER, J. Distributed objects for concurrent engineering. In *Proceedings of the 9th International Symposium on System Configuration Management* (London, UK, 1999), SCM-9, Springer-Verlag, pp. 172–185.
- [38] ESTUBLIER, J. Software configuration management: a roadmap. In *Proceedings of the Conference on The Future of Software Engineering* (New York, NY, USA, 2000), ICSE '00, ACM, pp. 279–289.
- [39] ESTUBLIER, J.; LEBLANG, D.; CLEMM, G.; CONRADI, R.; TICHY, W.; VAN DER HOEK, A.; WIBORG-WEBER, D. Impact of the research community on the field of software configuration management: summary of an impact project report. *SIGSOFT Softw. Eng. Notes* 27, 5 (Sept. 2002), 31–39.

- [40] ESTUBLIER, J.; LEBLANG, D.; HOEK, A. v. d.; CONRADI, R.; CLEMM, G.; TICHY, W.; WIBORG-WEBER, D. Impact of software engineering research on the practice of software configuration management. *ACM Trans. Softw. Eng. Methodol.* 14 (October 2005), 383–430.
- [41] FALLERI, J.-R.; TEYTON, C.; FOUCAULT, M.; PALYART, M.; MORANDAT, F.; BLANC, X. The harmony platform.
- [42] FECHNER, B. Gpu-based parallel signature scanning and hash generation. In *International Conference on Architecture of Computing Systems (ARCS)* (Feb 2010), pp. 1–6.
- [43] FEILER, P. H. Configuration management models in commercial environments. Technical Report 7 CMU/SEI-91-TR-7 ESD-9-TR-7, Software Engineering Institute, 1991.
- [44] FRITZ, T.; MURPHY, G. C. Using information fragments to answer the questions developers ask. In *ACM/IEEE International Conference on Software Engineering - Volume 1* (New York, NY, USA, 2010), ICSE '10, ACM, pp. 175–184.
- [45] GALL, H. C.; FLURI, B.; PINZGER, M. Change analysis with evolizer and changedistiller. *IEEE Softw.* 26, 1 (Jan. 2009), 26–33.
- [46] GIRBA, T.; KUHN, A.; SEEBERGER, M.; DUCASSE, S. How developers drive software evolution. In *International Workshop on Principles of Software Evolution* (Washington, DC, USA, 2005), IWPSE '05, IEEE Computer Society, pp. 113–122.
- [47] GIT-ANNEX. Git-annex, 2012. [Online; accessed 02 September 2014].
- [48] GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing*, 3 ed. Prentice Hall, 2008.
- [49] GRABLER, F.; AGRAWALA, M.; LI, W.; DONTCHEVA, M.; IGARASHI, T. Generating photo manipulation tutorials by demonstration. In *ACM SIGGRAPH 2009 Papers* (New York, NY, USA, 2009), SIGGRAPH '09, ACM, pp. 66:1–66:9.
- [50] GREEN, T.; PETRE, M. Usability analysis of visual programming environments: A ‘cognitive dimension’ framework. *Journal of Visual Languages & Computing* 7, 2 (1996), 131 – 174.
- [51] GROSSMAN, T.; MATEJKA, J.; FITZMAURICE, G. Chronicle: capture, exploration, and playback of document workflow histories. In *ACM symposium on User interface software and technology* (New York, NY, USA, 2010), UIST '10, ACM, pp. 143–152.
- [52] HAMMING, R. Error Detecting and Error Correcting Codes. *Bell System Technical Journal* 26, 2 (1950), 147–160.
- [53] HASS, A. M. J. *The Project Manager’s Guide to Software Engineering’s Best Practices*, 1 ed. IEEE Computer Society Press and John Wiley & Sons, 2002.
- [54] HASS, A. M. J. *Configuration Management Principles and Practice*. Pearson Education, Inc, Boston, MA, 2003.

- [55] HEER, J.; CARD, S. K.; LANDAY, J. A. prefuse: a toolkit for interactive information visualization. In *SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2005), CHI '05, ACM, pp. 421–430.
- [56] HERBSLEB, J. D.; GRINTER, R. E. Splitting the organization and integrating the code: Conway's law revisited. In *International Conference on Software Engineering* (New York, NY, USA, 1999), ICSE '99, ACM, pp. 85–95.
- [57] HIRSCHBERG, D. S. Algorithms for the longest common subsequence problem. *J. ACM* 24, 4 (Oct. 1977), 664–675.
- [58] HU, S.-M.; XU, K.; MA, L.-Q.; LIU, B.; JIANG, B.-Y.; WANG, J. Inverse image editing: Recovering a semantic editing history from a before-and-after image pair. *ACM Trans. Graph.* 32, 6 (Nov. 2013), 194:1–194:11.
- [59] HUNT, J. J.; VO, K.-P.; TICHY, W. F. Delta algorithms: an empirical analysis. *ACM Trans. Softw. Eng. Methodol.* 7, 2 (Apr. 1998), 192–214.
- [60] HUNT, J. W.; MCILROY, M. D. An algorithm for differential file comparison. Tech. rep., AT&T Bell Laboratories, Inc., 1975.
- [61] HUNT, J. W.; MCILROY, M. D. An algorithm for differential file comparison. Tech. Rep. CSTR 41, Bell Laboratories, Murray Hill, NJ, 1976.
- [62] JACKSON, D.; LADD, D. A. Semantic diff: A tool for summarizing the effects of modifications. In *International Conference on Software Maintenance* (Washington, DC, USA, 1994), ICSM '94, IEEE Computer Society, pp. 243–252.
- [63] JACOBSEN, J.; SCHLENKER, T.; EDWARDS, L. *Implementing a Digital Asset Management System: For Animation, Computer Games, and Web Development*. Focal Press, 2005.
- [64] JÚNIOR, J. R. D. S.; CLUA, E.; MURTA, L.; SARMA, A. Exploratory data analysis of software repositories via gpu processing. In *International Conference on Software Engineering and Knowledge Engineering, Hyatt Regency, Vancouver, BC, Canada, July 1-3, 2013*. (2014), pp. 495–500.
- [65] KAGDI, H.; POSHYVANYK, D. Who can help me with this change request? In *International Conference on Program Comprehension*. (May 2009), pp. 273–277.
- [66] KHAYAM, S. A. The discrete cosine transform (dct): Theory and application. department of electrical & computing engineering, 2003.
- [67] KLEMMER, S. R.; THOMSEN, M.; PHELPS-GOODMAN, E.; LEE, R.; LANDAY, J. A. Where do web sites come from?: capturing and interacting with design history. In *SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves* (New York, NY, USA, 2002), CHI '02, ACM, pp. 1–8.
- [68] KROGH, P. *The DAM Book: Digital Asset Management for Photographers*. O'Reilly, 2009.

- [69] KRÜGER, J.; WESTERMANN, R. Linear algebra operators for gpu implementation of numerical algorithms. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 908–916.
- [70] KURLANDER, D. Watch what i do. MIT Press, Cambridge, MA, USA, 1993, ch. Chimera: example-based graphical editing, pp. 271–290.
- [71] LAYERVAULT. Layervault, 2012. [Online; Accessed 02 September 2012].
- [72] LE GALL, D. Mpeg: a video compression standard for multimedia applications. *Commun. ACM* 34, 4 (Apr. 1991), 46–58.
- [73] LEON, A. *A Guide to Software Configuration Management*. Artech House Publishers, Norwood, MA, 2000.
- [74] LIN, C.-Y.; CHANG, S.-F. A robust image authentication method distinguishing jpeg compression from malicious manipulation. *Circuits and Systems for Video Technology, IEEE Transactions on* 11, 2 (Feb 2001), 153–168.
- [75] LIN, W.-Y.; TSENG, M.-C.; SU, J.-H. A confidence-lift support specification for interesting associations mining. In *Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining* (London, UK, UK, 2002), PAKDD '02, Springer-Verlag, pp. 148–158.
- [76] LINDHOLM, E.; NICKOLLS, J.; OBERMAN, S.; MONTRYM, J. Nvidia tesla: A unified graphics and computing architecture. *IEEE Micro* 28 (March 2008), 39–55.
- [77] LIOU, M. Overview of the p?64 kbit/s video coding standard. *Commun. ACM* 34, 4 (Apr. 1991), 59–63.
- [78] MAFRA, S. N.; TRAVASSOS, G. H. In *Simposio Brasileiro de Engenharia de Software* (2005), vol. 1, pp. 72–87.
- [79] McDONALD, D. W.; ACKERMAN, M. S. Expertise recommender: A flexible recommendation system and architecture. In *Computer Supported Cooperative Work* (New York, NY, USA, 2000), CSCW '00, ACM, pp. 231–240.
- [80] MENEELY, A.; WILLIAMS, L. Socio-technical developer networks: Should we trust our measurements? In *International Conference on Software Engineering* (New York, NY, USA, 2011), ICSE '11, ACM, pp. 281–290.
- [81] MENS, T. A state-of-the-art survey on software merging. *IEEE Trans. Softw. Eng.* 28, 5 (May 2002), 449–462.
- [82] MINTO, S.; MURPHY, G. C. Recommending emergent teams. In *International Workshop on Mining Software Repositories* (Washington, DC, USA, 2007), MSR '07, IEEE Computer Society, pp. 5–.
- [83] MONTONI, M. *Uma Abordagem para Condução de Iniciativas de Melhoria de Processos de Software*. Tese de Doutorado, COPPE/UFRJ, 2007.
- [84] MURTA, L.; OLIVEIRA, H.; DANTAS, C.; LOPES, L. G.; WERNER, C. Odyssey-scm: An integrated software configuration management infrastructure for uml models. *Sci. Comput. Program.* 65 (March 2007), 249–274.

- [85] MURTA, L. G. P. Version control - an introduction. University Lecture, August 2012.
- [86] MURTA, L. G. P. Uma introdução aos sistemas de controle de versão distribuídos. CBSOft Lecture, September 2014.
- [87] NICKOLLS, J.; BUCK, I.; GARLAND, M.; SKADRON, K. Scalable parallel programming with cuda. *Queue* 6 (March 2008), 40–53.
- [88] NVIDIA. Cuda zone. [Online; accessed 13 August 2010].
- [89] O’SULLIVAN, B. Making sense of revision-control systems. *Commun. ACM* 52, 9 (Sept. 2009), 56–62.
- [90] PARADOWSKI, M.; SLUZEK, A. Detection of image fragments related by affine transforms: Matching triangles and ellipses. In *International Conference on Information Science and Applications (ICISA)* (April 2010), pp. 1–8.
- [91] POSNETT, D.; D&#039;SOUZA, R.; DEVANBU, P.; FILKOV, V. Dual ecological measures of focus in software development. In *International Conference on Software Engineering* (Piscataway, NJ, USA, 2013), ICSE ’13, IEEE Press, pp. 452–461.
- [92] RIVEST, R. L. The MD5 Message-Digest Algorithm (RFC 1321). <http://www.ietf.org/rfc/rfc1321.txt?number=1321>.
- [93] SARMA, A.; MACCHERONE, L.; WAGSTROM, P.; HERBSLEB, J. Tesseract: Interactive visual exploration of socio-technical relationships in software development. In *International Conference on Software Engineering* (Washington, DC, USA, 2009), ICSE ’09, IEEE Computer Society, pp. 23–33.
- [94] SATISH, N.; HARRIS, M.; GARLAND, M. Designing efficient sorting algorithms for manycore gpus. In *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on* (May 2009), pp. 1–10.
- [95] SCHEFSTÖRM, D.; VAN DEN BROEK, G. *Tool Integration: Environments and Frameworks*. John Wiley & Sons, 1993.
- [96] SCHULER, D.; ZIMMERMANN, T. Mining usage expertise from version archives. In *International Working Conference on Mining Software Repositories* (New York, NY, USA, 2008), MSR ’08, ACM, pp. 121–124.
- [97] STEWARD, D. The design structure system: A method for managing the design of complex systems. *Engineering Management, IEEE Transactions on EM-28*, 3 (Aug 1981), 71–74.
- [98] STRANG, G. The discrete cosine transform. *SIAM Rev.* 41, 1 (Mar. 1999), 135–147.
- [99] SWAMINATHAN, A.; MAO, Y.; WU, M. Robust and secure image hashing. *Information Forensics and Security, IEEE Transactions on* 1, 2 (June 2006), 215–230.
- [100] SWIERSTRA, W.; LÖH, A. The semantics of version control. In *ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software* (New York, NY, USA, 2014), ACM, pp. 43–54.



- 
- [101] TICHY, W. F. Rcs - a system for version control. *Softw. Pract. Exper.* 15, 7 (July 1985), 637–654.
  - [102] WONG, J.; CAPRETZ, M. A. M. An implementation for merging images for version control. In *WSEAS international conference on Computers* (Stevens Point, Wisconsin, USA, 2006), ICCOMP'06, World Scientific and Engineering Academy and Society (WSEAS), pp. 662–667.
  - [103] ZAUNER, C. Implementation and Benchmarking of Perceptual Image Hash Functions. Master's thesis, University of Applied Sciences, Austria, 2010.
  - [104] ZIMMERMANN, T.; WEISGERBER, P.; DIEHL, S.; ZELLER, A. Mining version histories to guide software changes. In *International Conference on Software Engineering* (Washington, DC, USA, 2004), ICSE '04, IEEE Computer Society, pp. 563–572.

# APPENDIX A – Systematic mapping on version control over multimedia artifacts

## A.1 Introduction

Systematic mapping studies are defined in literature as a specific research methodology aimed at collecting and evaluating available evidences related to a subject [14]. According to Mafra and Travassos [78], a literature review without reproducibility property can be less embracing, not confident, and reviewer dependent. As a consequence, defining a research protocol can considerably increase the coverage and confidence during a literature reviewing process.

In this thesis we present a systematic mapping study aimed at locating and classifying related research to version control of multimedia artifact, specifically for images and videos. Using the systematic mapping technique and considering our experience in this field, we hope to reduce the risks presented by informal literature review. As a consequence, we expect to cover as many work as possible and allow the reproducibility of this review.

In order to perform this study, we define three activities, a proposed by Montoni [83]:

- *Protocol development*: define a research protocol aimed at guiding the study direction, test, and evaluate the protocol. The protocol is evaluated in order to check its viability as well as to make the necessary adjustments.
- *Research conduction*: the research is conducted using the protocol defined before and the results are evaluated. This activity also involves quantitative and qualitative analysis of the collected data.
- *Publish the results*: this process involves publish the results in a conference, magazine, or scientific library.

Due to the coverage of this work, systematic mapping study protocol includes image and video artifacts. In the following subsections, the study is detailed. We present the search protocol statement in Section A.2. Section A.3 presents how the search protocol is used. Section A.4 presents the results of our systematic mapping study. Finally, Section A.5 presents the complete list of analyzed papers.

## A.2 Systematic mapping protocol

In this section, we define a research protocol to identify and analyze existing works on versioning over multimedia artifacts (video and image). We first identify specific questions about how these artifacts have been processed, such as: techniques being applied, supporting tools, user intervention, among others.

Focusing on obtaining the desired information for this systematic mapping study, the main research question (MQ) and secondary questions (SQ) are the following:

**MQ:** How version control techniques are being used over multimedia artifacts?

**SQ1:** Which techniques are being used?

**SQ2:** Which tool support is offered?

**SQ3:** Does it integrates with existing VCS?

**SQ4:** What intervention degree is necessary by the user?

**SQ5:** How the described proposal has been evaluated?

Some criteria were established to guarantee the study viability (considering costs, efforts, and time), data accessibility, and comprehensiveness. This research was performed using digital libraries, through their own search mechanism. Manual search is done when data is unavailable to be collected by the aforementioned search mechanisms.

These libraries were selected by adopting the following criteria:

- Possess search mechanism, allowing logical expressions to be used;
- Retrieved control work previously obtained in an informal literature review;
- Possess search mechanisms that allow full text search, or at publications' specific fields;
- Guarantee unique results for the same search expression; and

- Include papers correlated with the research topic of this thesis in their database.

Due to the fact that almost all papers and conferences are only available in English, this idiom has been defined as default. However, papers in Portuguese from Brazilian conference were also considered.

The following digital libraries have been selected:

- *IeeeXplorer* (IEEE) (in mode *Command Search*): <http://ieeexplorer.ieee.org>
- *Scopus* (in mode *Advanced Search*): <http://www.scopus.com>

These libraries have been selected due to their easy access for retrieving references and the full paper. Additionally, these libraries achieved good results in other work [83] [10]. Besides, they have been classified as relevant sources in Software Engineering. The ACM (*Association for Computing Machinery*) library was not used because the same query does not produces the same results for different searches. The manual search considered both forward and backward snow bowling to analyze the cited references of papers considered in this work.

The following sections present the search methodology and protocol evaluation as well as the criteria used for papers selection. Additionally, we also present extraction and data storage.

### A.2.1 Search methodology and protocol evaluation

In order to define the search expression to be used over the digital libraries, first a test and refinement process has been established. In this process, population, intervention, comparison, and output parameters have been determined.

**(P) (Population):** image and video artifacts.

**(I) (Intervention):** versioning techniques.

**(C) (Comparison):** do not exist as the main objective is not to compare approaches but to characterize them.

**(O) (Output):** tools, methodologies, and approaches.

After the protocol and parameters definition have been established, the next step was to scope the search expression to populate, or instantiate, the output elements. As a result, the expression used to perform the search was defined as following:

**Population:** (*“image” OR “images” OR “picture” OR “pictures” OR “video” OR “videos” OR “movie” OR “movies”*). Some terms have been include during the tests, as some papers use different names to represent the same concept.

**Intervention:** (*“version control” OR “revision control” OR “conflict resolution” OR “system” OR “control”*). Here the term *merge* and *diff* have been omitted due to the amount of papers that use them without any relation to software engineering.

**Comparison:** no applicable.

**Output:** (*“tool” OR “editor” OR “prototype” OR “methodology” OR “procedure” OR “algorithm” OR “infrastructure”*).

The third step involves identifying control papers, aiming at validating the protocol. In this case, a relevant paper in the context of version control for image and video was used:

- Hsiang-Ting Chen, Li-Yi Wei, and Chun-Fa Chang. 2011. Nonlinear revision control for images. In ACM SIGGRAPH 2011 papers (SIGGRAPH '11), Hugues Hoppe (Ed.). ACM, New York, NY, USA, , Article 105 , pp. 105:1–105:10.

The next step comprises the validation of the search protocol using the control paper. At first, the query was submitted to *Scopus* in order to fine tune the search protocol. In sequence, the other search engine was used with the search protocol already tuned. During the validation of the search expression, we could observe that terms such as “system” and “control” returned a high number of non-related papers to the subject of this study. This mainly happens because they are very common to others knowledge areas. To avoid this problem, these terms were removed.

Finally, after the search expression adjustments, the control paper was returned, thus validating the search expression. The final expression used at *Scopus* and all others search mechanisms is defined as:

*TITLE-ABS-KEY((“image” OR “images” OR “picture” OR “pictures” OR “video” OR “videos” OR “movie” OR “movies”) AND (“project management” OR “version control” OR “revision control” OR “conflict resolution”) AND (“tool” OR “editor” OR “prototype” OR “methodology” OR “procedure” OR “algorithm” OR “infrastructure”)) AND (LIMIT-TO(SUBJAREA, “COMP” ))*

The same *Scopus* search string was used at *IEEEExplore*, but removing the beginning of the string (*“TITLE-ABS-KEY”*), which is responsible to search in title, abstract, and

key words fields, and the final of the string (*“AND ( LIMIT-TO( SUBJAREA,“COMP” ) )”*), responsible to limit the knowledge area. The search was made only over metadata (title, abstract, and key words).

### A.2.2 Selection criteria

The selection of papers is done in three steps:

**1st Step - preliminary selection:** it was done by using the search expression in the digital libraries or through manual search. The resulting papers were cataloged to a future analysis and exclusion filter processing. Table A.1 lists the returned results of this step.

**2nd Step - relevant publications selection (1o filter):** the search mechanism used does not guarantee that all results are pertinent to this thesis subject, as just syntactic restrictions exist. As a consequence, a manual filter was applied by reading and analyzing the title and abstract of each paper returned in the 1st step. Paper exclusion was based on the following criteria:

- EC1 - Papers that do not deal with versioning of multimedia artifacts (images and videos);
- EC2 - Papers qualified as norm, templates, or national and international patterns;
- EC3 - Papers that described keynote speeches, tutorials, courses, workshops, and so on;
- EC4 – Papers not available in full version at digital libraries nor any other way without costs to the researcher; and
- EC5 – Papers not written in English or Portuguese.

The selected papers to the next step were the ones not excluded by any of these filters.

**3rd Step - relevant publications selection (2o filter):** after the first filter, we still have no guarantee that the remaining papers are tied to this thesis research as just paper’s title and abstract were manually analyzed. The next step involves a complete reading and further analysis of individual papers selected in the second step, and applying the same exclusion criteria stated before.

### A.2.3 Extraction and data storage procedure

Each paper retrieved by the search expression in the 1st step has been stored. For each publication stored, the filters defined in the 2nd and 3rd step were applied, being identified by “EC[exclusion criteria number]” or “OK” in Table A.1. Additionally, this table also identifies the artifact type discussed in the respective paper (image or video).

Finally, each publication selected on the 3rd step had its reference data completely registered, such as abstract, observations, and the answer for each of the secondary questions previously defined. All these articles are presented in Section A.6.

## A.3 Search procedure

After definition and adjustment in the search protocol, the process was executed between December 2012 and February 2013 (updated in August 2015), using the *Scopus* and *IEEEExplore* digital libraries. The digital libraries returned a total of 414 distinct papers. The manual search were also performed, with a selection of 2 papers, according to Figure A.1.

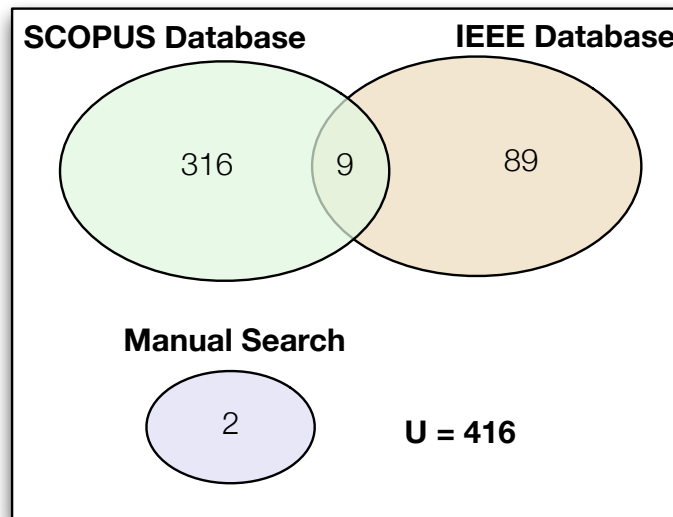


Figure A.1: Papers returned by the digital library.

The second step to select the papers was performed by reading their title and abstracts. This way, following the established criteria, 10 papers were selected from the search result. Figure A.2 shows the distribution of these 10 publications in addition to the 2 papers selected manually.

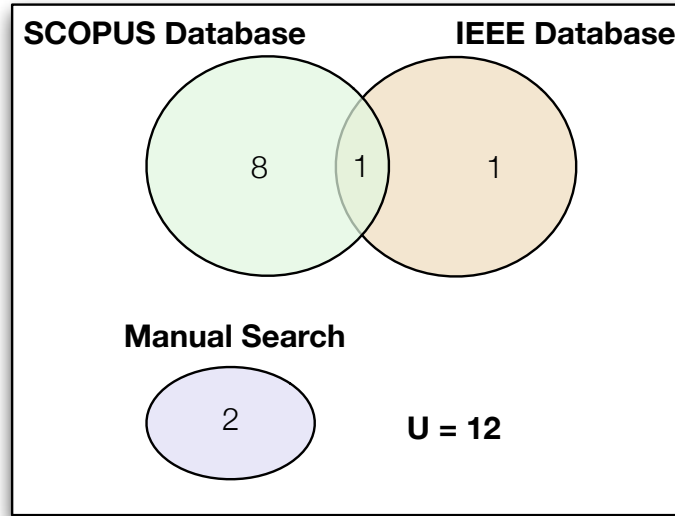


Figure A.2: Papers selected through the first filter.

From all papers returned by the digital libraries, 2 could not be considered due to their unavailability. A contact with their authors was performed without success.

Publications that passed through the 1st and 2nd steps were read and classified again in relation to the same exclusion criteria, but now considering their complete text. During the 3rd step execution, 4 publications from the search result were selected, classified in relation to their artifact type, summarized, and had the secondary questions answered. Considering the publications that were approved in the 2nd filter, we obtained the distribution presented in Figure A.3.

In Figure A.3, it is possible to observe that by just considering the digital libraries, the *Scopus* library represents about 100% of results while *IEEE* library represents 25%.



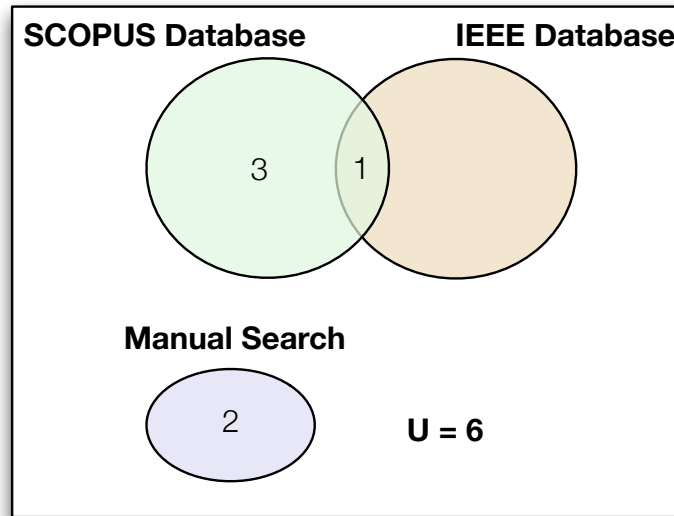


Figure A.3: Papers selected through the second filter.

## A.4 Results analysis

This section presents an analysis over the consolidate data related to papers that were considered tied to this thesis research theme. Figure A.4 illustrates the papers' distribution according to artifact type. Details of each publication can be visualized on Section A.6.

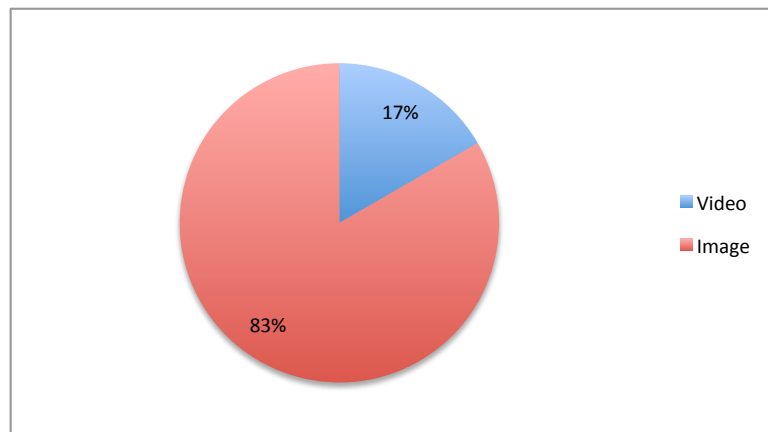


Figure A.4: Paper classification by multimedia artifacts.

Considering SQ1 (*Which techniques are being used?*), we aimed to evaluate the methodology used for version control over multimedia artifacts. These results are grouped according to techniques used by each paper in Figure A.5. Many approaches use techniques based on file system and graph.

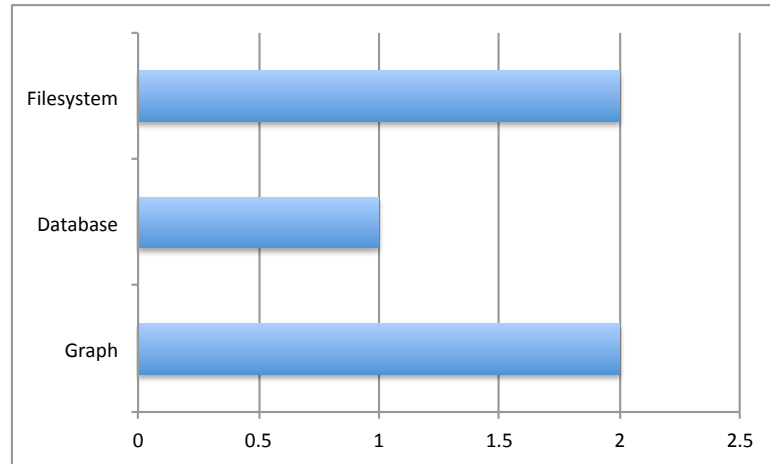


Figure A.5: Techniques used for multimedia artifact management.

Following, SQ2 (*Which tool support is offered?*), we evaluated the results considering what was mentioned in the paper in relation to tools support. The results of this step can be seen in Figure A.6, showing that half of approaches have some kind of tool for supporting the user.

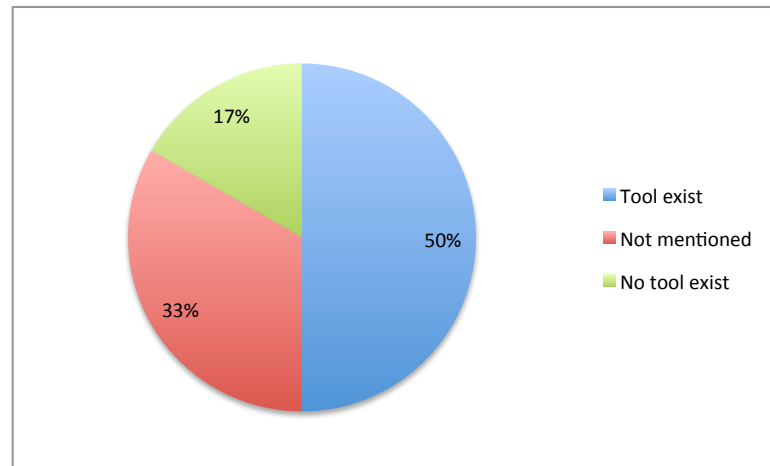


Figure A.6: Tool support offered.

SQ3 (*Does it integrates with existing VCS?*) is aimed to check the availability of integration of the proposed approach and a VCS, such as Git, Mercurial, or SVN. One important fact that can be seen in Figure A.7 is that just 33% of the approaches offer such kind of integration. This factor can be considered a negative aspect for projects that already use a VCS on a daily basis.

When checking SQ4 (*What intervention degree is necessary by the user?*), it is possible to check information upon the necessity of the user intervention degree to perform a *diff*, *patch*, and *merge* operations over artifacts. According to Figure A.8, about 33% of the

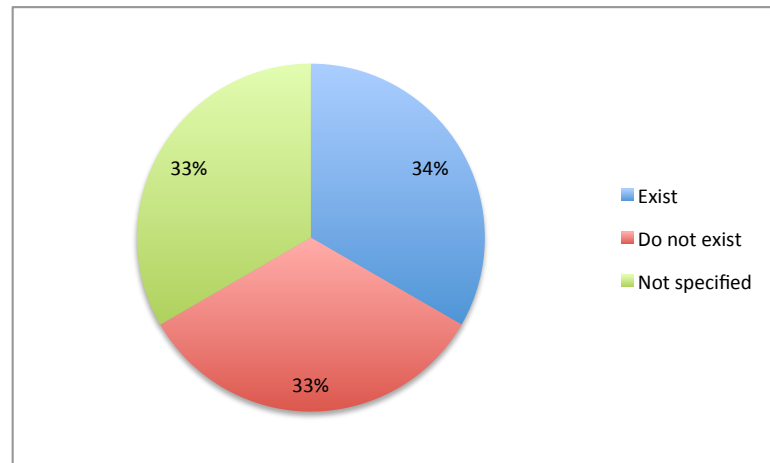
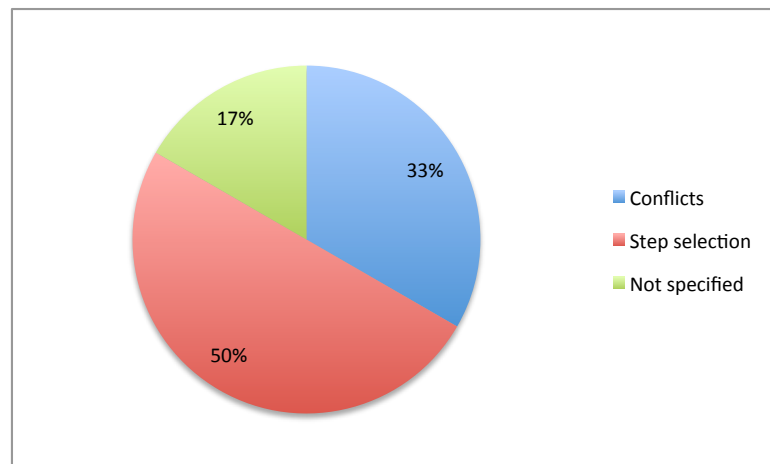


Figure A.7: VCS integration availability.

approaches require user intervention just in conflict situations. Besides that, in 50% of the approaches, the user is responsible to specify the *diff*, *patch*, and *merge* processing order.

Figure A.8: User intervention degree necessary for applying *diff*, *patch*, and *merge* operations over artifacts.

Finally, SQ5 (*How the described proposal has been evaluated?*) gives information about how the proposed approach was evaluated. Figure A.9 shows that most of papers are evaluated by analyzing storage space and processing time. It is important to state that papers that evaluate more than one item are considered twice in this analysis.

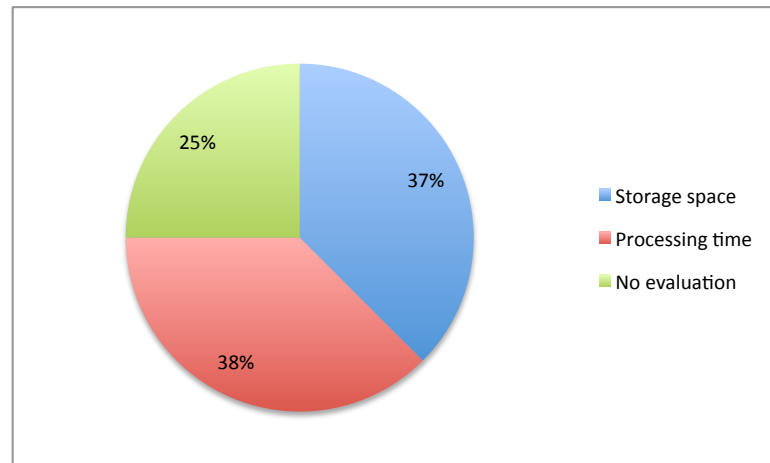


Figure A.9: Dependent variables used to evaluate the paper.

## A.5 Complete list of papers returned by the search expression

Table A.1: Complete list returned by systematic mapping review.

Title	Year	Complete Reference	1st Filter	2nd Filter	Source	Artifact
Image-based localization and content authoring in structure-from-motion point cloud models for real-time field reporting applications	2015	Bae H., Golparvar-Fard M., White J.. Image-based localization and content authoring in structure-from-motion point cloud models for real-time field reporting applications. Journal of Computing in Civil Engineering, 2015 pp - .	EC1	-	SCOPUS	-
A little ingenuity solves an elephant-sized problem	2015	Byrd G.. A little ingenuity solves an elephant-sized problem. Computer, 2015 pp 74 - 77.	EC1	-	SCOPUS	-
Efficient image-aware version control systems using GPU	2015	da Silva Junior J.R., Clua E., Murta L.. Efficient image-aware version control systems using GPU. Software - Practice and Experience, 2015 pp - .	OK	OK	SCOPUS	Image
CVC-FP and SGT: a new database for structural floor plan analysis and its groundtruthing tool	2015	de las Heras L.-P., Terrades O.R., Robles S., Sanchez G.. CVC-FP and SGT: a new database for structural floor plan analysis and its groundtruthing tool. International Journal on Document Analysis and Recognition, 2015 pp 15 - 30.	EC1	-	SCOPUS	-
Negotiation Strategy Video Modeling Training for Adolescents with Autism Spectrum Disorder: A Usability Study	2015	Hochhauser M., Gal E., Weiss P.L.. Negotiation Strategy Video Modeling Training for Adolescents with Autism Spectrum Disorder: A Usability Study. International Journal of Human-Computer Interaction, 2015 pp 472 - 480.	EC1	-	SCOPUS	-
A framework for model-driven acquisition and analytics of visual data using UAVs for automated construction progress monitoring	2015	Lin J.J., Han K.K., Golparvar-Fard M.. A framework for model-driven acquisition and analytics of visual data using UAVs for automated construction progress monitoring. Congress on Computing in Civil Engineering, Proceedings, 2015 pp 156 - 164.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

Status quo and open challenges in vision-based sensing and tracking of temporary resources on infrastructure construction sites	2015	Teizer J.. Status quo and open challenges in vision-based sensing and tracking of temporary resources on infrastructure construction sites. <i>Advanced Engineering Informatics</i> , 2015 pp 225 - 238.	EC1	-	SCOPUS	-
Construction performance monitoring via still images, time-lapse photos, and video streams: Now, tomorrow, and the future	2015	Yang J., Park M.-W., Vela P.A., Golparvar-Fard M.. Construction performance monitoring via still images, time-lapse photos, and video streams: Now, tomorrow, and the future. <i>Advanced Engineering Informatics</i> , 2015 pp 211 - 224.	EC1	-	SCOPUS	-
System model of an image stabilization system	2014	Carmona M., Gomez J.M., Roma D., Casas A., Lopez M., Bosch J., Herms A., Sabater J., Volkmer R., Heidecke F., Maue T., Nakai E., Schmidt W.. System model of an image stabilization system. <i>Proceedings of SPIE - The International Society for Optical Engineering</i> , 2014 pp - .	EC1	-	SCOPUS	-
Fast updating of national geo-spatial databases with high resolution imagery: China's methodology and experiences	2014	Chen J., Wang D., Zhao R., Zhang H., Liao A., Liu J.. Fast updating of national geo-spatial databases with high resolution imagery: China's methodology and experiences. <i>International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives</i> , 2014 pp 41 - 50.	EC1	-	SCOPUS	-
An end-to-end simulation framework for the Large Synoptic Survey Telescope	2014	Connolly A.J., Angeli G.Z., Chandrasekharan S., Claver C.F., Cook K., Ivezić Z., Jones R.L., Krughoff K.S., Peng E.-H., Peterson J., Petry C., Rasmussen A.P., Ridgway S.T., Saha A., Sembroski G., Vanderplas J., Yoachim P.. An end-to-end simulation framework for the Large Synoptic Survey Telescope. <i>Proceedings of SPIE - The International Society for Optical Engineering</i> , 2014 pp - .	EC1	-	SCOPUS	-
Serious sustainability challenge game to promote teaching and learning of building sustainability	2014	Dib H., Adamo-Villani N.. Serious sustainability challenge game to promote teaching and learning of building sustainability. <i>Journal of Computing in Civil Engineering</i> , 2014 pp - .	EC1	-	SCOPUS	-
The realization of the geological image 3D reconstruction	2014	Du C., Leng B.. The realization of the geological image 3D reconstruction. <i>2014 International Conference on Mechatronics, Electronic, Industrial and Control Engineering, MEIC 2014</i> , 2014 pp 14 - 18.	EC1	-	SCOPUS	-
Pythy: Improving the introductory Python programming experience	2014	Edwards S.H., Tilden D.S., Allevato A.. Pythy: Improving the introductory Python programming experience. <i>SIGCSE 2014 - Proceedings of the 45th ACM Technical Symposium on Computer Science Education</i> , 2014 pp 641 - 646.	EC1	-	SCOPUS	-
Model-based quality management of software development projects	2014	Heidrich J., Rombach D., Klas M.. Model-based quality management of software development projects. <i>Software Project Management in a Changing World</i> , 2014 pp 125 - 156.	EC1	-	SCOPUS	-
An evidence based approach for multiple similarity measures combining for ontology mapping	2014	Idoudi R., Ettabaa K.S., Hamrouni K., Solaiman B.. An evidence based approach for multiple similarity measures combining for ontology mapping. <i>International Image Processing, Applications and Systems Conference, IPAS 2014</i> , 2014 pp - .	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

Wind responses of Giant Magellan telescope	2014	Irrazaval B., Buleri C., Johns M.. Wind responses of Giant Magellan telescope. Proceedings of SPIE - The International Society for Optical Engineering, 2014 pp - .	EC1	-	SCOPUS	-
ConstructAide: Analyzing and visualizing construction sites through photographs and building models	2014	Karsch K., Golparvar-Fard M., Forsyth D.. ConstructAide: Analyzing and visualizing construction sites through photographs and building models. ACM Transactions on Graphics, 2014 pp - .	EC1	-	SCOPUS	-
From big data to big projects: A step-by-step roadmap	2014	Mousannif H., Sabah H., Douiji Y., Sayad Y.O.. From big data to big projects: A step-by-step roadmap. Proceedings - 2014 International Conference on Future Internet of Things and Cloud, FiCloud 2014, 2014 pp 373 - 378.	EC1	-	SCOPUS	-
E-ELT requirements management	2014	Schneller D.. E-ELT requirements management. Proceedings of SPIE - The International Society for Optical Engineering, 2014 pp - .	EC1	-	SCOPUS	-
Root cause analysis towards lean collaboration between production line and factory planning	2014	Slitnikova S., Fruchter R.. Root cause analysis towards lean collaboration between production line and factory planning. Computing in Civil and Building Engineering - Proceedings of the 2014 International Conference on Computing in Civil and Building Engineering, 2014 pp 1449 - 1456.	EC1	-	SCOPUS	-
Complexity in the MATISSE cold optics: A risk or a tool?	2014	Tromp N., Bettonvil F., Aitink-Kroes G., Agocs T., Navarro R.. Complexity in the MATISSE cold optics: A risk or a tool?. Proceedings of SPIE - The International Society for Optical Engineering, 2014 pp - .	EC1	-	SCOPUS	-
The tail wags the dog: Managing large telescope construction projects with lagging requirements and creeping scope	2014	Warner M.. The tail wags the dog: Managing large telescope construction projects with lagging requirements and creeping scope. Proceedings of SPIE - The International Society for Optical Engineering, 2014 pp - .	EC1	-	SCOPUS	-
TOAD: A numerical model for the 4MOST instrument	2014	Winkler R., Haynes D.M., Bellido-Tirado O., Xu W., Haynes R.. TOAD: A numerical model for the 4MOST instrument. Proceedings of SPIE - The International Society for Optical Engineering, 2014 pp - .	EC1	-	SCOPUS	-
Object tracking in video sequences using information fusion principles. Meanshift kernel implementation using fuzzy rules	2013	Alam I.. Object tracking in video sequences using information fusion principles. Meanshift kernel implementation using fuzzy rules. 2013 5th Computer Science and Electronic Engineering Conference, CEEC 2013 - Conference Proceedings, 2013 pp 146 - 151.	EC1	-	SCOPUS	-
Alignment of large project management process to business strategy: A review and conceptual framework	2013	Alsudiri T., Al-Karaghoul W., Eldabi T.. Alignment of large project management process to business strategy: A review and conceptual framework. Journal of Enterprise Information Management, 2013 pp 596 - 615.	EC1	-	SCOPUS	-
Social networking meets software development: Perspectives from git hub, MSDN, stack exchange, and top coder	2013	Begel A., Bosch J., Storey M.-A.. Social networking meets software development: Perspectives from git hub, MSDN, stack exchange, and top coder. IEEE Software, 2013 pp 52 - 66.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

The role of software in environmental conflict resolution: How did MarineMap facilitate collaborative learning in california's MLPA initiative?	2013	Cravens A.E.. The role of software in environmental conflict resolution: How did MarineMap facilitate collaborative learning in california's MLPA initiative?. Computer-Supported Collaborative Learning Conference, CSCL, 2013 pp 468 - .	EC1	-	SCOPUS	-
Leveraging transparency	2013	Dabbish L., Stuart C., Tsay J., Herbsleb J.. Leveraging transparency. IEEE Software, 2013 pp 37 - 43.	EC1	-	SCOPUS	-
Authoring multi-stage code examples with editable code histories	2013	Ginosar S., De Pombo L.F., Agrawala M., Hartmann B.. Authoring multi-stage code examples with editable code histories. UIST 2013 - Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology, 2013 pp 485 - 494.	EC1	-	SCOPUS	-
A Petri Net based algorithm for the resource constrained project scheduling problem (RCPSP): A real life application in the Animation and Videogame industry	2013	Mejia G., Sanchez M.A., Nino K., Figueroa P.. A Petri Net based algorithm for the resource constrained project scheduling problem (RCPSP): A real life application in the Animation and Videogame industry. 22nd International Conference on Production Research, ICPR 2013, 2013 pp - .	EC1	-	SCOPUS	-
Do we need total quality management in fusion engineering? - Experience from construction of W7-X	2013	Vilbrandt R., Bosch H.-S., Feist J.-H.. Do we need total quality management in fusion engineering? - Experience from construction of W7-X. 2013 IEEE 25th Symposium on Fusion Engineering, SOFE 2013, 2013 pp - .	EC1	-	SCOPUS	-
Error-correction methods for construction site image processing under changing illumination conditions	2013	Wu Y., Kim C., Kim H.. Error-correction methods for construction site image processing under changing illumination conditions. Journal of Computing in Civil Engineering, 2013 pp 99 - 109.	EC1	-	SCOPUS	-
Fostering a continuum of care	2012	Bonfiglio S.. Fostering a continuum of care. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2012 pp 35 - 41.	EC1	-	SCOPUS	-
Smarter financial management - An answer to the missing link between key performance indicators and budgeting decisions for smarter cities	2012	Boyette N., Fang H.. Smarter financial management - An answer to the missing link between key performance indicators and budgeting decisions for smarter cities. Annual SRII Global Conference, SRII, 2012 pp 608 - 613.	EC1	-	SCOPUS	-
A GPU-based architecture for parallel image-aware version control	2012	Da Silva Jr. J.R., Pacheco T., Clua E., Murta L.. A GPU-based architecture for parallel image-aware version control. Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR, 2012 pp 191 - 200.	OK	OK	SCOPUS, IEEE	Image
Quality evaluation of information technology project - A multicriteria approach	2012	E Silva L.C., Costa A.P.C.S.. Quality evaluation of information technology project - A multicriteria approach. Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics, 2012 pp 611 - 616.	EC1	-	SCOPUS	-
RFID and CCTV-based material delivery monitoring for cable-stayed bridge construction	2012	Ju Y., Kim C., Kim H.. RFID and CCTV-based material delivery monitoring for cable-stayed bridge construction. Journal of Computing in Civil Engineering, 2012 pp 183 - 190.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

The large synoptic survey telescope Project Management Control System	2012	Kantor J.P.. The large synoptic survey telescope Project Management Control System. Proceedings of SPIE - The International Society for Optical Engineering, 2012 pp - .	EC1	-	SCOPUS	-
Investigating the effect of software project type on accuracy of software development effort estimation in COCOMO model	2012	Khatibi B V., Khatibi E.. Investigating the effect of software project type on accuracy of software development effort estimation in COCOMO model. Proceedings of SPIE - The International Society for Optical Engineering, 2012 pp - .	EC1	-	SCOPUS	-
Linked open data aggregation: Conflict resolution and aggregate quality	2012	Knap T., Michelfeit J., Necasky M.. Linked open data aggregation: Conflict resolution and aggregate quality. Proceedings - International Computer Software and Applications Conference, 2012 pp 106 - 111.	EC1	-	IEEE, SCOPUS	-
Statelets: Coordination of social collaboration processes	2012	Liptchinsky V., Khazankin R., Truong H.-L., Dustdar S.. Statelets: Coordination of social collaboration processes. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2012 pp 1 - 16.	EC1	-	SCOPUS	-
Product and quality assurance processes and ECSS compliance within a science ground segment using Gaia as an example	2012	Lock T., Mercier E., Els S., Gracia G., O'Mullane W., Comoretto G., Gallegos J.. Product and quality assurance processes and ECSS compliance within a science ground segment using Gaia as an example. Proceedings of SPIE - The International Society for Optical Engineering, 2012 pp - .	EC1	-	SCOPUS	-
MESSI, the METIS instrument software simulator	2012	Nicolini G., Andretta V., Abbo L., Antonucci E., Bemporad A., Capobianco G., Crescenzo G., Fineschi S., Focardi M., Magli E., Naletto G., Nicolosi G., Pancrazzi M., Ricci M., Romoli M., Uslenghi M., Volpicelli A.. MESSI, the METIS instrument software simulator. Proceedings of SPIE - The International Society for Optical Engineering, 2012 pp - .	EC1	-	SCOPUS	-
An empirical study of application of PSP methodology with students of a Systems Technology program with different levels of training	2012	Nino Manrique J.F., Anaya R.. An empirical study of application of PSP methodology with students of a Systems Technology program with different levels of training. 38th Latin America Conference on Informatics, CLEI 2012 - Conference Proceedings, 2012 pp - .	EC1	-	SCOPUS	-
Integrated telescope model for the James Webb space telescope	2012	Scott Knight J., Scott Acton D., Lightsey P., Barto A.. Integrated telescope model for the James Webb space telescope. Proceedings of SPIE - The International Society for Optical Engineering, 2012 pp - .	EC1	-	SCOPUS	-
PIVoT: Project insights and Visualization Toolkit	2012	Sharma V.S., Kaulgud V.. PIVoT: Project insights and Visualization Toolkit. 2012 3rd International Workshop on Emerging Trends in Software Metrics, WETSoM 2012 - Proceedings, 2012 pp 63 - 69.	EC1	-	SCOPUS	-
A holistic approach to developing a progress tracking system for distributed agile teams	2012	Sultan A., Ivins W.K., Gray W.A.. A holistic approach to developing a progress tracking system for distributed agile teams. Proceedings - 2012 IEEE/ACIS 11th International Conference on Computer and Information Science, ICIS 2012, 2012 pp 503 - 512.	EC1	-	SCOPUS	-
Local thermal seeing modeling validation through observatory measurements	2012	Vogiatzis K., Otarola A., Skidmore W., Travouillon T., Angeli G.. Local thermal seeing modeling validation through observatory measurements. Proceedings of SPIE - The International Society for Optical Engineering, 2012 pp - .	EC1	-	SCOPUS	-



Table A.1: Complete list returned by systematic mapping review.

Six Sigma in IT processes, IT services and IT products - A fact or a fad? Six Sigma beyond manufacturing in IT processes, IT services and IT products	2012	Wong W.Y., Lee C.W., Tshai K.Y.. Six Sigma in IT processes, IT services and IT products - A fact or a fad? Six Sigma beyond manufacturing in IT processes, IT services and IT products. Proceedings - 2012 IEEE 12th International Conference on Computer and Information Technology, CIT 2012, 2012 pp 524 - 531.	EC1	-	SCOPUS	-
The similarity degree comparison study of travel routes between the VR digital models and the reality environment in urban planning	2012	Zhang Y., Zhang X., Chen Y.. The similarity degree comparison study of travel routes between the VR digital models and the reality environment in urban planning. Journal of Convergence Information Technology, 2012 pp 116 - 123.	EC1	-	SCOPUS	-
DVB-T2 LDPC decoder with perfect conflict resolution	2012	Xiongxin Zhao; Zhixiang Chen; Xiao Peng; Dajiang Zhou; Goto, S.; , DVB-T2 LDPC decoder with perfect conflict resolution, VLSI Design, Automation, and Test (VLSI-DAT), 2012 International Symposium on , vol., no., pp.1-4, 23-25 April 2012	EC1	-	IEEE	-
Efficient Versioning for Scientific Array Databases	2012	Seering, A.; Cudre-Mauroux, P.; Madden, S.; Stonebraker, M.; , Efficient Versioning for Scientific Array Databases, Data Engineering (ICDE), 2012 IEEE 28th International Conference on , vol., no., pp.1013-1024, 1-5 April 2012	EC1	-	IEEE	-
Industrialised Building System (IBS) in Sarawak construction industry	2012	Bohari, A.A.M.; Mahat, N.; Kipli, K.; , Industrialised Building System (IBS) in Sarawak construction industry, Innovation Management and Technology Research (ICIMTR), 2012 International Conference on , vol., no., pp.433-437, 21-22 May 2012	EC1	-	IEEE	-
Software development process animation	2011	Agarwal R.. Software development process animation. Proceedings of the Annual Southeast Conference, 2011 pp 221 - 226.	EC1	-	SCOPUS	-
Dubaisat-1: Mission overview and applications	2011	Alrais A., Alsuwaidi A., Bushahab A.. Dubaisat-1: Mission overview and applications. 2011 IEEE GCC Conference and Exhibition, GCC 2011, 2011 pp 65 - 66.	EC1	-	SCOPUS	-
Real options pricing by the finite element method	2011	Andalaft-Chacur A., Montaz Ali M., Gonzalez Salazar J.. Real options pricing by the finite element method. Computers and Mathematics with Applications, 2011 pp 2863 - 2873.	EC1	-	SCOPUS	-
Nonlinear revision control for images	2011	Chen H.-T., Wei L.-Y., Chang C.-F.. Nonlinear revision control for images. ACM Transactions on Graphics, 2011 pp - .	OK	OK	SCOPUS	Image
Defect estimation using capture-recapture in IBM Jazz	2011	Doran J., Gary K.. Defect estimation using capture-recapture in IBM Jazz. Proceedings of the IASTED International Conference on Software Engineering and Applications, SEA 2011, 2011 pp 176 - 183.	EC1	-	SCOPUS	-
A visual monitoring framework for integrated productivity and carbon footprint control of construction operations	2011	Heydarian A., Golparvar-Fard M.. A visual monitoring framework for integrated productivity and carbon footprint control of construction operations. Congress on Computing in Civil Engineering, Proceedings, 2011 pp 504 - 511.	EC1	-	SCOPUS	-
Immersive and non immersive 3D virtual city: Decision support tool for urban sustainability	2011	Isaacs J.P., Gilmour D.J., Blackwood D.J., Falconer R.E.. Immersive and non immersive 3D virtual city: Decision support tool for urban sustainability. Electronic Journal of Information Technology in Construction, 2011 pp 151 - 162.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

Vehicle detection using partial least squares	2011	Kembhavi A., Harwood D., Davis L.S.. Vehicle detection using partial least squares. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011 pp 1250 - 1265.	EC1	-	SCOPUS	-
A pilot study of a 3D game environment for construction safety education	2011	Lin K.-Y., Son J.W., Rojas E.M.. A pilot study of a 3D game environment for construction safety education. Electronic Journal of Information Technology in Construction, 2011 pp 69 - 83.	EC1	-	SCOPUS	-
Use of SAR data for hydro-morphological characterization in sub-Saharan Africa: A case study	2011	Papa M.N., Ciervo F., Koussoube Y., Di Martino G., Iodice A., Riccio D., Ruello G., Zinno I.. Use of SAR data for hydro-morphological characterization in sub-Saharan Africa: A case study. Proceedings of SPIE - The International Society for Optical Engineering, 2011 pp - .	EC1	-	SCOPUS	-
DubaiSat-1 mission overview	2011	Rais A.A.. DubaiSat-1 mission overview. Proceedings of SPIE - The International Society for Optical Engineering, 2011 pp - .	EC1	-	SCOPUS	-
The RECorder: A participatory digital platform for the rehabilitation of the city of Famagusta in Cyprus	2011	Vardouli T., Xanthouli E., Stathopoulos A.. The RECorder: A participatory digital platform for the rehabilitation of the city of Famagusta in Cyprus. Proceedings - 2011 7th International Conference on Intelligent Environments, IE 2011, 2011 pp 238 - 244.	EC1	-	IEEE, SCOPUS	-
Leveraging cloud platform for custom application development	2011	Zhou N., An D.P., Zhang L.-J., Wong C.-H.. Leveraging cloud platform for custom application development. Proceedings - 2011 IEEE International Conference on Services Computing, SCC 2011, 2011 pp 584 - 591.	EC1	-	IEEE, SCOPUS	-
A Decision Support System for Global Software Development	2011	Beecham, S.; Noll, J.; Richardson, I.; Dhungana, D.; , A Decision Support System for Global Software Development, Global Software Engineering Workshop (ICGSEW), 2011 Sixth IEEE International Conference on , vol., no., pp.48-53, 15-18 Aug. 2011	EC1	-	IEEE	-
Advances space technology for project management in early stage of the design work	2011	Kurnaz, S.; Rustamov, R.B.; Hasanova, S.N.; Rahimov, N.; , Advances space technology for project management in early stage of the design work, Recent Advances in Space Technologies (RAST), 2011 5th International Conference on , vol., no., pp.128-131, 9-11 June 2011	EC1	-	IEEE	-
Application of separation principle in Management innovation based on TRIZ	2011	Yaqiang Zhang; Xiufeng Sang; , Application of separation principle in Management innovation based on TRIZ, Computer Science and Service System (CSSS), 2011 International Conference on , vol., no., pp.2116-2119, 27-29 June 2011	EC1	-	IEEE	-
Goal-Driven Development Method for Managing Embedded System Projects: An Industrial Experience Report	2011	Guoping Rong; Dong Shao; He Zhang; Jun Li; , Goal-Driven Development Method for Managing Embedded System Projects: An Industrial Experience Report, Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on , vol., no., pp.414-423, 22-23 Sept. 2011	EC1	-	IEEE	-
The development of the enterprise innovation value diagnosis system with the use of systems engineering	2011	Wang, T.J.; Chang, L.; , The development of the enterprise innovation value diagnosis system with the use of systems engineering, System Science and Engineering (ICSSE), 2011 International Conference on , vol., no., pp.373-378, 8-10 June 2011	EC1	-	IEEE	-

Table A.1: Complete list returned by systematic mapping review.

Visualizing Work Progress Information of Construction Project by Web and VR Application	2011	Leen-Seok Kang; Hyoun-Seok Moon; Hyun-Seoung Kim; Gwang-Ryul Choi; Nam-Jin Park; Chang-Hak Kim; , Visualizing Work Progress Information of Construction Project by Web and VR Application, Modelling Symposium (AMS), 2011 Fifth Asia , vol., no., pp.177-180, 24-26 May 2011	EC1	-	IEEE	-
100 Million hours of audiovisual content: Digital preservation and access in the PrestoPRIME project	2010	Addis M., Allasia W., Bailer W., Boch L., Gallo F., Wright R.. 100 Million hours of audiovisual content: Digital preservation and access in the PrestoPRIME project. ACM International Conference Proceeding Series, 2010 pp - .	EC1	-	SCOPUS	-
Safe and flexible human-robot cooperation in industrial applications	2010	Bosch J.J., Klett F.. Safe and flexible human-robot cooperation in industrial applications. 2010 International Conference on Computer Information Systems and Industrial Management Applications, CISIM 2010, 2010 pp 107 - 110.	EC1	-	SCOPUS	-
Scheduling piece requests blindly and randomly for peer-to-peer live streaming	2010	Chen Y.-S., Chen C.-J., Zhao Y.-X., Li C.-X.. Scheduling piece requests blindly and randomly for peer-to-peer live streaming. Journal of China Universities of Posts and Telecommunications, 2010 pp 76 - 84.	EC1	-	SCOPUS	-
A novel method for de-warping in Persian document images captured by cameras	2010	Dehbovid H., Razzazi F., Alirezaii S.. A novel method for de-warping in Persian document images captured by cameras. 2010 International Conference on Computer Information Systems and Industrial Management Applications, CISIM 2010, 2010 pp 614 - 619.	EC1	-	SCOPUS	-
Automatic MPEG4 compatible face representation using clustering-based modeling schemes	2010	Ghahari A., Mosleh M.. Automatic MPEG4 compatible face representation using clustering-based modeling schemes. 2010 International Conference on Computer Information Systems and Industrial Management Applications, CISIM 2010, 2010 pp 96 - 102.	EC1	-	SCOPUS	-
Hybrid clustering-based 3D face modeling upon non-perfect orthogonality of frontal and profile views	2010	Ghahari A., Mosleh M.. Hybrid clustering-based 3D face modeling upon non-perfect orthogonality of frontal and profile views. 2010 International Conference on Computer Information Systems and Industrial Management Applications, CISIM 2010, 2010 pp 578 - 584.	EC1	-	SCOPUS	-
Computer vision-based video interpretation model for automated productivity analysis of construction operations	2010	Gong J., Caldas C.H.. Computer vision-based video interpretation model for automated productivity analysis of construction operations. Journal of Computing in Civil Engineering, 2010 pp 252 - 263.	EC1	-	SCOPUS	-
Integration of different computational models in a computer vision framework	2010	Kasprzak W.. Integration of different computational models in a computer vision framework. 2010 International Conference on Computer Information Systems and Industrial Management Applications, CISIM 2010, 2010 pp 13 - 18.	EC1	-	SCOPUS	-
SOFIA telescope modal survey test and test-model correlation	2010	Keas P., Brewster R., Guerra J., Lampater U., Karcher H., Teufel S., Wagner J.. SOFIA telescope modal survey test and test-model correlation. Proceedings of SPIE - The International Society for Optical Engineering, 2010 pp - .	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

Development and clinical evaluation of a physiological data acquisition device for monitoring and exercise guidance of heart failure and chronic heart disease patients	2010	Kokonozzi A., Astaras A., Semertzidis P., Michail E., Filos D., Chouvarda I., Grossenbacher O., Koller J.-M., Leopoldo R., Porchet J.-A., Corveon M., Luprano J., Sipila A., Zamboulis C., Maglaveras N.. Development and clinical evaluation of a physiological data acquisition device for monitoring and exercise guidance of heart failure and chronic heart disease patients. Computing in Cardiology, 2010 pp 1099 - 1102.	EC1	-	SCOPUS	-
The large synoptic survey telescope preliminary design overview	2010	Krabbendam V.L., Sweeney D.. The large synoptic survey telescope preliminary design overview. Proceedings of SPIE - The International Society for Optical Engineering, 2010 pp - .	EC1	-	SCOPUS	-
Phase retrieval analysis of the Hobby-Eberly Telescope primary mirror segment figure error and its implication for wavefront sensing for the new wide-field upgrade	2010	Lee H., Hill G.J., Hart M.. Phase retrieval analysis of the Hobby-Eberly Telescope primary mirror segment figure error and its implication for wavefront sensing for the new wide-field upgrade. Proceedings of SPIE - The International Society for Optical Engineering, 2010 pp - .	EC1	-	SCOPUS	-
Investigation of disturbance effects on space-based weak lensing measurements with an integrated model	2010	Lieber M., Kaplan M., Sholl M., Bernstein G.. Investigation of disturbance effects on space-based weak lensing measurements with an integrated model. Proceedings of SPIE - The International Society for Optical Engineering, 2010 pp - .	EC1	-	SCOPUS	-
CMYK model color image segmentation using type 2 fuzzy sets	2010	Maity S., Sil J.. CMYK model color image segmentation using type 2 fuzzy sets. 2010 International Conference on Computer Information Systems and Industrial Management Applications, CISIM 2010, 2010 pp 347 - 352.	EC1	-	SCOPUS	-
Documentary tools in everyday life: The wedding planner	2010	McKenzie P.J., Davies E.. Documentary tools in everyday life: The wedding planner. Journal of Documentation, 2010 pp 788 - 806.	EC1	-	SCOPUS	-
Transcription support system using Subversion	2010	Murakawa T., Fukuoka H., Noda D., Nakagawa M.. Transcription support system using Subversion. WEBIST 2010 - Proceedings of the 6th International Conference on Web Information Systems and Technology, 2010 pp 150 - 155.	EC1	-	SCOPUS	-
Pruned-AZB for reduced complexity block matching in video compression	2010	Pandit A.K., Verma S., Tomar G.S.. Pruned-AZB for reduced complexity block matching in video compression. 2010 International Conference on Computer Information Systems and Industrial Management Applications, CISIM 2010, 2010 pp 553 - 556.	EC1	-	SCOPUS	-
Integrated finite element analysis and raytracing, oriented to structural optimization, for astronomical instrument design	2010	Riva M., De Caprio V., Spano P., Tintori M.. Integrated finite element analysis and raytracing, oriented to structural optimization, for astronomical instrument design. Proceedings of SPIE - The International Society for Optical Engineering, 2010 pp - .	EC1	-	SCOPUS	-
Effects of thermal deformations on the sensitivity of optical systems for space application	2010	Segato E., Da Deppo V., Debei S., Cremonese G., Cherubini G.. Effects of thermal deformations on the sensitivity of optical systems for space application. Proceedings of SPIE - The International Society for Optical Engineering, 2010 pp - .	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

Professional Penetration Testing	2010	Wilhelm T.. Professional Penetration Testing. Professional Penetration Testing, 2010 pp - .	EC1	-	SCOPUS	-
Scholarly knowledge development and dissemination in an international context: Approaches and tools for higher education	2010	Willis J., Baron J., Lee R.-A., Gozza-Cohen M., Currie A.. Scholarly knowledge development and dissemination in an international context: Approaches and tools for higher education. Computers in the Schools, 2010 pp 155 - 199.	EC1	-	SCOPUS	-
Object recognition in construction-site images using 3D CAD-based filtering	2010	Wu Y., Kim H., Kim C., Han S.H.. Object recognition in construction-site images using 3D CAD-based filtering. Journal of Computing in Civil Engineering, 2010 pp 56 - 64.	EC1	-	SCOPUS	-
A Collaborative Platform for Sharing Scientific Data and Experiments: Application to Characterization Experiments of Photovoltaic Cells	2010	Kossi, T.; Fazziki, A.E.; Napo, K.; , A Collaborative Platform for Sharing Scientific Data and Experiments: Application to Characterization Experiments of Photovoltaic Cells, Signal-Image Technology and Internet-Based Systems (SITIS), 2010 Sixth International Conference on , vol., no., pp.329-335, 15-18 Dec. 2010	EC1	-	IEEE	-
Data conflict resolution for layered LDPC decoding algorithm by selective recalculation	2010	Wen Ji; Hamaminato, M.; Nakayama, H.; Goto, S.; , Data conflict resolution for layered LDPC decoding algorithm by selective recalculation, Image and Signal Processing (CISP), 2010 3rd International Congress on , vol.6, no., pp.2985-2989, 16-18 Oct. 2010	EC1	-	IEEE	-
Establishing the Impact Evaluation Indicators System for Rural Road Investment Projects: Evidence from Fujian Province, China	2010	Chen Yuefeng; Tian Yuan; Chen Xiaohong; , Establishing the Impact Evaluation Indicators System for Rural Road Investment Projects: Evidence from Fujian Province, China, Optoelectronics and Image Processing (ICOIP), 2010 International Conference on , vol.1, no., pp.492-497, 11-12 Nov. 2010	EC1	-	IEEE	-
Information system for managing the ionizing radiations based medical procedures and the patient dose RXINFO	2010	Badoiu, Adrian; Petrescu, Sanda; Botu, Alexandru; Vlad, Vasilica; Matei, Gheorghe; , Information system for managing the ionizing radiations based medical procedures and the patient dose — RXINFO, Automation Quality and Testing Robotics (AQTR), 2010 IEEE International Conference on , vol.2, no., pp.1-4, 28-30 May 2010	EC1	-	IEEE	-
Software tool for supporting ethnographic research in design and innovation projects in management education	2010	Agrawal, M.; Agarwal, A.; krishnamoorthy, a.; Pendse, P.; , Software tool for supporting ethnographic research in design and innovation projects in management education, Technology for Education (T4E), 2010 International Conference on , vol., no., pp.63-67, 1-3 July 2010	EC1	-	IEEE	-
Direct automatic generation of mind maps from text with M2Gen	2009	Abdeen M., El-Sahan R., Ismaeil A., El-Harouny S., Shalaby M., Yagoub M.C.E.. Direct automatic generation of mind maps from text with M2Gen. TIC-STH'09: 2009 IEEE Toronto International Conference - Science and Technology for Humanity, 2009 pp 95 - 99.	EC1	-	SCOPUS	-
DUBAISAT-1: Mission overview, development status and future applications	2009	Al Rais A.A., Al Suwaidi A., Ghedira H.. DUBAISAT-1: Mission overview, development status and future applications. International Geoscience and Remote Sensing Symposium (IGARSS), 2009 pp V196 - V199.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

A methodology of requirement analysis for disaster response oriented spatial information	2009	Biao C., Jinggao Z., Jun L., Zuo Z.. A methodology of requirement analysis for disaster response oriented spatial information. 2009 WASE International Conference on Information Engineering, ICIE 2009, 2009 pp 341 - 345.	EC1	-	SCOPUS	-
The implementation of satellite images and associated digital image processing in addition to GIS modelling for urban mapping in Amman area, Jordan	2009	Fadda E.H.R., Kakish M., Al Azab T.A.. The implementation of satellite images and associated digital image processing in addition to GIS modelling for urban mapping in Amman area, Jordan. WSEAS Transactions on Communications, 2009 pp 300 - 309.	EC1	-	SCOPUS	-
Towards automated progress assessment of workpackage components in construction projects using computer vision	2009	Ibrahim Y.M., Lukins T.C., Zhang X., Trucco E., Kaka A.P.. Towards automated progress assessment of workpackage components in construction projects using computer vision. Advanced Engineering Informatics, 2009 pp 93 - 103.	EC1	-	SCOPUS	-
Software engineering challenges in game development	2009	Kanode C.M., Haddad H.M.. Software engineering challenges in game development. ITNG 2009 - 6th International Conference on Information Technology: New Generations, 2009 pp 260 - 265.	OK	EC1	SCOPUS	-
Dialogue games that agents play within a society	2009	Karunatilake N.C., Jennings N.R., Rahwan I., McBurney P.. Dialogue games that agents play within a society. Artificial Intelligence, 2009 pp 935 - 981.	EC1	-	SCOPUS	-
GIMIAS: An open source framework for efficient development of research tools and clinical prototypes	2009	Larrabide I., Omedas P., Martelli Y., Planes X., Nieber M., Moya J.A., Butakoff C., Sebastian R., Camara O., De Craene M., Bijmens B.H., Frangi A.F.. GIMIAS: An open source framework for efficient development of research tools and clinical prototypes. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2009 pp 417 - 426.	EC1	-	SCOPUS	-
A multi-agent system to construct production orders by employing an expert system and a neural network	2009	Lopez-Ortega O., Villar-Medina I.. A multi-agent system to construct production orders by employing an expert system and a neural network. Expert Systems with Applications, 2009 pp 2937 - 2946.	EC1	-	SCOPUS	-
DAM good: Making the most of your assets in a multimedia world	2009	McClure M.. DAM good: Making the most of your assets in a multimedia world. EContent, 2009 pp 28 - 32.	OK	EC4	SCOPUS	Video
Validation tool for 2D multi-stage metal-forming processes on meta-stable stainless steels	2009	Post J., de Vries C., Huetink J.. Validation tool for 2D multi-stage metal-forming processes on meta-stable stainless steels. Journal of Materials Processing Technology, 2009 pp 5558 - 5572.	EC1	-	SCOPUS	-
Heading into new virtual environments: What skills do design team members need?	2009	Sher W., Sherratt S., Williams A., Gameson R.. Heading into new virtual environments: What skills do design team members need?. Electronic Journal of Information Technology in Construction, 2009 pp 17 - 29.	EC1	-	SCOPUS	-
Job security	2009	Spinellis D.. Job security. IEEE Software, 2009 pp 14 - 15.	EC1	-	SCOPUS	-
Evaluating the Quality of Open Source Software	2009	Spinellis D., Gousios G., Karakoidas V., Louridas P., Adams P.J., Samoladas I., Stamelos I.. Evaluating the Quality of Open Source Software. Electronic Notes in Theoretical Computer Science, 2009 pp 5 - 28.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

A simplified approach to determine airspace complexity maps under automated conflict resolution	2009	Salaun, E.; Vela, A.E.; Feron, E.; Clarke, J.-P.; Solak, S.; , A simplified approach to determine airspace complexity maps under automated conflict resolution, Digital Avionics Systems Conference, 2009. DASC '09. IEEE/AIAA 28th , vol., no., pp.3.C.5-1-3.C.5-13, 23-29 Oct. 2009	EC1	-	IEEE	-
A Specialized Meta-scheduler for Business and Applications Constraints Management	2009	Chevalier, J.; Mouton, S.; , A Specialized Meta-scheduler for Business and Applications Constraints Management, Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International , vol.2, no., pp.365-370, 20-24 July 2009	EC1	-	IEEE	-
Biomedical data acquisition and processing in the Decision Support services of HEARTFAID platform	2009	Costanzo, D.; , Biomedical data acquisition and processing in the Decision Support services of HEARTFAID platform, Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2009. IDAACS 2009. IEEE International Workshop on , vol., no., pp.292-296, 21-23 Sept. 2009	EC1	-	IEEE	-
Conflict resolution for pipelined layered LDPC decoders	2009	Marchand, C.; Dore, J.-B.; Conde-Canencia, L.; Boutillon, E.; , Conflict resolution for pipelined layered LDPC decoders, Signal Processing Systems, 2009. SiPS 2009. IEEE Workshop on , vol., no., pp.220-225, 7-9 Oct. 2009	EC1	-	IEEE	-
Direct automatic generation of mind maps from text with M <sup>2</sup> Gen	2009	Abdeen, M.; El-Sahan, R.; Ismaeil, A.; El-Harouny, S.; Shalaby, M.; Yagoub, M.C.E.; , Direct automatic generation of mind maps from text with M2Gen, Science and Technology for Humanity (TIC-STH), 2009 IEEE Toronto International Conference , vol., no., pp.95-99, 26-27 Sept. 2009	EC1	-	IEEE	-
Implementation of power managed hyper transport system for transmission of HD video	2009	Kodati, A.V.; Vemuri, K.S.; Lili He; Jones, M.; , Implementation of power managed hyper transport system for transmission of HD video, Quality of Electronic Design, 2009. ISQED 2009. Quality Electronic Design , vol., no., pp.517-521, 16-18 March 2009	EC1	-	IEEE	-
Target Recognition Based on a Novel Riemannian Map	2009	Guangwei Li; Yunpeng Liu; Zelin Shi; Jian Yin; , Target Recognition Based on a Novel Riemannian Map, Image and Signal Processing, 2009. CISP '09. 2nd International Congress on , vol., no., pp.1-5, 17-19 Oct. 2009	EC1	-	IEEE	-
Workload Point System Based on Project Schedule Optimization	2009	Sun Yanan; Cui Rong; , Workload Point System Based on Project Schedule Optimization, Management and Service Science, 2009. MASS '09. International Conference on , vol., no., pp.1-4, 20-22 Sept. 2009	EC1	-	IEEE	-
Technology interactions on reticle delivery	2008	Ackmann P., Goad S., West C.. Technology interactions on reticle delivery. Proceedings of SPIE - The International Society for Optical Engineering, 2008 pp - .	EC1	-	SCOPUS	-
Systems engineering for the preliminary design of the thirty meter telescope	2008	Angeli G.Z., Roberts S., Vogiatzis K.. Systems engineering for the preliminary design of the thirty meter telescope. Proceedings of SPIE - The International Society for Optical Engineering, 2008 pp - .	EC1	-	SCOPUS	-
Computational models of the human body for medical image analysis	2008	Ayache N.. Computational models of the human body for medical image analysis. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2008 pp 405 - .	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

Kohonen map combined to the K-means algorithm for the identification of day types of Algerian electricity load	2008	Benabbas F., Khadir M.T., Fay D., Boughrira A.. Kohonen map combined to the K-means algorithm for the identification of day types of Algerian electricity load. Proceedings - 7th Computer Information Systems and Industrial Management Applications, CISIM 2008, 2008 pp 78 - 83.	EC1	-	SCOPUS	-
Autonomous high dynamic range phase unwrapping	2008	Bikkannavar S.. Autonomous high dynamic range phase unwrapping. Proceedings of SPIE - The International Society for Optical Engineering, 2008 pp - .	EC1	-	SCOPUS	-
Fairness assessment of the adaptive token bank fair queuing scheduling algorithm	2008	Bokhari F.A., Yanikomeroglu H., Wong W.K., Rahman M.. Fairness assessment of the adaptive token bank fair queuing scheduling algorithm. IEEE Vehicular Technology Conference, 2008 pp - .	EC1	-	SCOPUS	-
Applying a video-based requirements engineering technique to an airport scenario	2008	Bruegge B., Creighton O., Reiss M., Stangl H.. Applying a video-based requirements engineering technique to an airport scenario. 2008 3rd International Workshop on Multimedia and Enjoyable Requirements Engineering, MERE'08, 2008 pp - .	EC1	-	SCOPUS	-
The potential of crowd simulations for communication purposes in architecture	2008	Burkhard R., Bischof S., Herzog A.. The potential of crowd simulations for communication purposes in architecture. Proceedings of the International Conference on Information Visualisation, 2008 pp 403 - 408.	EC1	-	SCOPUS	-
An integrated FEM and ANN methodology for metal-formed product design	2008	Chan W.L., Fu M.W., Lu J.. An integrated FEM and ANN methodology for metal-formed product design. Engineering Applications of Artificial Intelligence, 2008 pp 1170 - 1181.	EC1	-	SCOPUS	-
Implicit personal contracts and actor-group consensus in CRM implementations - Evidence for their role in influencing success	2008	Corner I., Hinton M.. Implicit personal contracts and actor-group consensus in CRM implementations - Evidence for their role in influencing success. 2nd European Conference on Information Management and Evaluation, ECIME 2008, 2008 pp 89 - 98.	EC1	-	SCOPUS	-
SISI project: Developing GIS-based tools for vulnerability assessment	2008	Della Rocca B., Fattoruso G., Locurzio S., Pasanisi F., Pica R., Peloso A., Pollino M., Tebano C., Trocciola A., De Chiara D., Tortora G.. SISI project: Developing GIS-based tools for vulnerability assessment. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2008 pp 327 - 330.	EC1	-	SCOPUS	-
Integrated modeling for the GPi flexure sensitive structure	2008	Erickson D., Roberts S., Pazder J.S., Fletcher J.M.. Integrated modeling for the GPi flexure sensitive structure. Proceedings of SPIE - The International Society for Optical Engineering, 2008 pp - .	EC1	-	SCOPUS	-
An algorithm for parking lot occupation detection	2008	Fabian T.. An algorithm for parking lot occupation detection. Proceedings - 7th Computer Information Systems and Industrial Management Applications, CISIM 2008, 2008 pp 165 - 170.	EC1	-	SCOPUS	-
Multi-sensor-based fully autonomous non-cooperative collision avoidance system for unmanned air vehicles	2008	Fasano G., Accardo D., Moccia A., Carbone G., Ciniglio U., Corrado F., Luongo S.. Multi-sensor-based fully autonomous non-cooperative collision avoidance system for unmanned air vehicles. Journal of Aerospace Computing, Information and Communication, 2008 pp 338 - 360.	EC1	-	SCOPUS	-



Table A.1: Complete list returned by systematic mapping review.

The new CIO: from technician to business strategist and the implications for e-commerce	2008	Fortino A.. The new CIO: from technician to business strategist and the implications for e-commerce. IEEE International Conference on e-Business Engineering, ICEBE'08 - Workshops: AiR'08, EM2I'08, SOAIC'08, SOKM'08, BIMA'08, DKEEE'08, 2008 pp 139 - 146.	EC1	-	SCOPUS	-
Lean development in the automotive industry: The snap-shot approach	2008	Gracbsch M., Roclofscn J., Lindcmann U.. Lean development in the automotive industry: The snap-shot approach. FISITA World Automotive Congress 2008, Congress Proceedings - Mobility Concepts, Man Machine Interface, Process Challenges, Virtual Reality, 2008 pp 430 - 439.	EC1	-	SCOPUS	-
Exploring the 3D spatial distribution of cultural prints using remote sensing and giscience: Istanbul: The city on seven hills	2008	Has Y.. Exploring the 3D spatial distribution of cultural prints using remote sensing and giscience: Istanbul: The city on seven hills. International Geoscience and Remote Sensing Symposium (IGARSS), 2008 pp IV703 - IV706.	EC1	-	SCOPUS	-
ATST systems engineering - Project update and lessons learned	2008	Hubbard R.P.. ATST systems engineering - Project update and lessons learned. Proceedings of SPIE - The International Society for Optical Engineering, 2008 pp - .	EC1	-	SCOPUS	-
Performance prediction for a code with data-dependent runtimes	2008	Jarvis S.A., Foley B.P., Isitt P.J., Spooner D.P., Rueckert D., Nudd G.R.. Performance prediction for a code with data-dependent runtimes. Concurrency Computation Practice and Experience, 2008 pp 195 - 206.	EC1	-	SCOPUS	-
Application of support vector machine based on rough sets to project risk assessment (RS-SVM)	2008	Jia Z., Gong L., Han J.. Application of support vector machine based on rough sets to project risk assessment (RS-SVM). Proceedings - International Conference on Computer Science and Software Engineering, CSSE 2008, 2008 pp 508 - 511.	EC1	-	SCOPUS	-
Microarray image converted database - Genetic Algorithm application in bioinformatics	2008	Jiao C.Y., Li D.G.. Microarray image converted database - Genetic Algorithm application in bioinformatics. BioMedical Engineering and Informatics: New Development and the Future - Proceedings of the 1st International Conference on BioMedical Engineering and Informatics, BMEI 2008, 2008 pp 302 - 305.	EC1	-	SCOPUS	-
A web-based collaborative environment based on a shared ontology for the maintenance of steam turbines	2008	Khadir M.T., Sellami M.. A web-based collaborative environment based on a shared ontology for the maintenance of steam turbines. Proceedings - 7th Computer Information Systems and Industrial Management Applications, CISIM 2008, 2008 pp 151 - 152.	EC1	-	SCOPUS	-
Skeletal curves of 3D astrocyte samples	2008	Klette G.. Skeletal curves of 3D astrocyte samples. Machine Graphics and Vision, 2008 pp 105 - 129.	EC1	-	SCOPUS	-
A novel approach based on support vector machine to forecasting the construction project cost	2008	Kong F., Wu X.-J., Cai L.-Y.. A novel approach based on support vector machine to forecasting the construction project cost. Proceedings of the 2008 International Symposium on Computational Intelligence and Design, ISCID 2008, 2008 pp 21 - 24.	EC1	-	SCOPUS	-
Rough set approach for feature reduction in pattern recognition through unsupervised artificial neural network	2008	Kothari A.G., Keskar A.G., Gokhale A.P., Deshpande R., Deshmukh P.. Rough set approach for feature reduction in pattern recognition through unsupervised artificial neural network. Proceedings - 1st International Conference on Emerging Trends in Engineering and Technology, ICETET 2008, 2008 pp 1196 - 1199.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

The second life client-viewer: A case study in using open source	2008	Krawczyk L., Hansen S., Deshpande Y.. The second life client-viewer: A case study in using open source. AusWeb 2008: 14th Australasian World Wide Web Conference, 2008 pp - .	EC1	-	SCOPUS	-
WiiArts: Creating collaborative art experience with WiiRemote interaction	2008	Lee H.-J., Kim H., Gupta G., Mazalek A.. WiiArts: Creating collaborative art experience with WiiRemote interaction. TEF'08 - Second International Conference on Tangible and Embedded Interaction - Conference Proceedings, 2008 pp 33 - 36.	EC1	-	SCOPUS	-
Application and evaluation of wireless transmission technique at construction job-site	2008	Lin L.-K., Tuan C.-C., Cnen C.-J.. Application and evaluation of wireless transmission technique at construction job-site. Proceedings of the 3rd IEEE Asia-Pacific Services Computing Conference, APSCC 2008, 2008 pp 963 - 968.	EC1	-	SCOPUS	-
Adaptive critic learning techniques for engine torque and air-fuel ratio control	2008	Liu D., Javaherian H., Kovalenko O., Huang T.. Adaptive critic learning techniques for engine torque and air-fuel ratio control. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2008 pp 988 - 993.	EC1	-	SCOPUS	-
Object localization based on mutual information in global structure constraint model	2008	Liu M., Guo D., Jie M., Zhou C.. Object localization based on mutual information in global structure constraint model. Proceedings - 7th Computer Information Systems and Industrial Management Applications, CISIM 2008, 2008 pp 212 - 213.	EC1	-	SCOPUS	-
Conflict resolution within multi-agent system in collaborative design	2008	Liu Q., Cui X., Hu X.. Conflict resolution within multi-agent system in collaborative design. Proceedings - International Conference on Computer Science and Software Engineering, CSSE 2008, 2008 pp 520 - 523.	EC1	-	IEEE, SCOPUS	-
Application of a web-based education system in industrial processes	2008	Marino P., Dominguez M.A., Otero S., Merino M.. Application of a web-based education system in industrial processes. WEBIST 2008 - 4th International Conference on Web Information Systems and Technologies, Proceedings, 2008 pp 452 - 455.	EC1	-	SCOPUS	-
University-enterprise technology transfer for education and training about industrial processes	2008	Marino P., Dominguez M.A., Otero S., Merino M.. University-enterprise technology transfer for education and training about industrial processes. 2008 Conference on Human System Interaction, HSI 2008, 2008 pp 40 - 43.	EC1	-	IEEE, SCOPUS	-
Multimedia tool to help small and medium enterprises in their enterprise resource planning and business process change	2008	Marino P., Dominguez M.A., Otero S., Merino M.. Multimedia tool to help small and medium enterprises in their enterprise resource planning and business process change. Proceedings of the 2008 International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government, EEE 2008, 2008 pp 252 - 256.	EC1	-	SCOPUS	-
Classification of fMRI time series in a low-dimensional subspace with a spatial prior	2008	Meyer F.G., Shen X.. Classification of fMRI time series in a low-dimensional subspace with a spatial prior. IEEE Transactions on Medical Imaging, 2008 pp 87 - 98.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

High-resolution optical modeling of the thirty meter telescope for systematic performance trades	2008	Nissly C., Seo B.-J., Troy M., Angeli G., Angione J., Crossfield I., Ellerbroek B., Gilles L., Sigrist N.. High-resolution optical modeling of the thirty meter telescope for systematic performance trades. Proceedings of SPIE - The International Society for Optical Engineering, 2008 pp - .	EC1	-	SCOPUS	-
Two hand tracking using colour statistical model with the K-means embedded particle filter for hand gesture recognition	2008	Ongkittikul S., Worrall S., Kondoz A.. Two hand tracking using colour statistical model with the K-means embedded particle filter for hand gesture recognition. Proceedings - 7th Computer Information Systems and Industrial Management Applications, CISIM 2008, 2008 pp 201 - 206.	EC1	-	SCOPUS	-
Dome and mirror seeing estimates for the thirty meter telescope	2008	Pazder J.S., Vogiatzis K., Angeli G.Z.. Dome and mirror seeing estimates for the thirty meter telescope. Proceedings of SPIE - The International Society for Optical Engineering, 2008 pp - .	EC1	-	SCOPUS	-
Evolution of computer graphics and its impact on engineering product development	2008	Sathyanarayana K., Ravi Kumar G.V.V.. Evolution of computer graphics and its impact on engineering product development. Proceedings - Computer Graphics, Imaging and Visualisation, Modern Techniques and Applications, CGIV, 2008 pp 32 - 37.	EC1	-	SCOPUS	-
Optical vortex and correlation image sensor for networked deformation sensing of infrastructures	2008	Sato S., Kurihara T., Ando S., Fujimoto I.. Optical vortex and correlation image sensor for networked deformation sensing of infrastructures. Proceedings of INSS 2008 - 5th International Conference on Networked Sensing Systems, 2008 pp 39 - 42.	EC1	-	SCOPUS	-
Effect of similarity metrics and ROI sizes in featureless computer aided detection of breast masses in tomosynthesis	2008	Singh S., Tourassi G.D., Lo J.Y.. Effect of similarity metrics and ROI sizes in featureless computer aided detection of breast masses in tomosynthesis. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2008 pp 286 - 291.	EC1	-	SCOPUS	-
A segmentation-based approach for temporal analysis of software version repositories	2008	Siy H., Chundi P., Rosenkrantz D.J., Subramaniam M.. A segmentation-based approach for temporal analysis of software version repositories. Journal of Software Maintenance and Evolution, 2008 pp 199 - 222.	EC1	-	SCOPUS	-
Recent results obtained on the APEX 12 m antenna with the ArTeMiS prototype camera	2008	Talvard M., Andre P., Rodriguez L., Le-Pennec Y., De Breuck C., Reveret V., Agnese P., Boulade O., Doumayrou E., Dubreuil D., Ercolani E., Gallais P., Horeau B., Lagage P.O., Leriche B., Lortholary M., Martignac J., Minier V., Pantin E., Rabanus D., Relland J., Willmann G.. Recent results obtained on the APEX 12 m antenna with the ArTeMiS prototype camera. Proceedings of SPIE - The International Society for Optical Engineering, 2008 pp - .	EC1	-	SCOPUS	-
Towards a conceptual framework and tool support for linking long-term product and business planning with agile software development	2008	Vahaniitty J., Rautiainen K.T.. Towards a conceptual framework and tool support for linking long-term product and business planning with agile software development. Proceedings - International Conference on Software Engineering, 2008 pp 25 - 28.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

Meteorological simulation portal on the thaigrid system	2008	Vittayasermsthan S., Wongjampa K., Yamwong W., Sarochawikast R.. Meteorological simulation portal on the thaigrid system. 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, ECTI-CON 2008, 2008 pp 153 - 156.	EC1	-	SCOPUS	-
Conditional correlation analysis for safe region-based memory management	2008	Wang X., Xu Z., Liu X., Guo Z., Wang X., Zhang Z.. Conditional correlation analysis for safe region-based memory management. Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), 2008 pp 45 - 55.	EC1	-	SCOPUS	-
Influence of media communication on daily communication	2008	Yamada-Kawai K., Musou M., Hirasawa N.. Influence of media communication on daily communication. MCCSIS'08 - IADIS Multi Conference on Computer Science and Information Systems; Proceedings of ICT, Society and Human Beings 2008, 2008 pp 192 - 196.	EC1	-	SCOPUS	-
Priority assessing method for aviation multi-project based on fuzzy comprehensive evaluation	2008	Yu J., Zhang J., Li Y.. Priority assessing method for aviation multi-project based on fuzzy comprehensive evaluation. 2008 International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2008, 2008 pp - .	EC1	-	SCOPUS	-
Intercity commute patterns in Central Texas	2008	Zhan F.B., Chen X.. Intercity commute patterns in Central Texas. Proceedings of SPIE - The International Society for Optical Engineering, 2008 pp - .	EC1	-	SCOPUS	-
Using the CAT for 3D sketching in front of large displays	2008	Zhang H., Hadim J., Granier X.. Using the CAT for 3D sketching in front of large displays. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2008 pp 8 - 19.	EC1	-	SCOPUS	-
Application of geo-spatial information technology in the engineering manage of roller compaction construction	2008	Zhang J., Chen X., Zhong C., Wu H., Duan S.. Application of geo-spatial information technology in the engineering manage of roller compaction construction. International Geoscience and Remote Sensing Symposium (IGARSS), 2008 pp III1312 - III1315.	EC1	-	SCOPUS	-
3-D axon structure extraction and analysis in confocal fluorescence microscopy images	2008	Zhang Y., Zhou X., Lu J., Lichtman J., Adjeroh D., Wong S.T.. 3-D axon structure extraction and analysis in confocal fluorescence microscopy images. 2007 IEEE/NIH Life Science Systems and Applications Workshop, LISA, 2008 pp 241 - 244.	EC1	-	SCOPUS	-
Applying SE methods achieves project success to evaluate hammer and fixed cutter grinders using multiple varieties and moistures of biomass feedstock for ethanol Production	2008	Zirker L.R., Wright C.T., Hamelin R.D.. Applying SE methods achieves project success to evaluate hammer and fixed cutter grinders using multiple varieties and moistures of biomass feedstock for ethanol Production. 18th Annual International Symposium of the International Council on Systems Engineering, INCOSE 2008, 2008 pp 2357 - 2368.	EC1	-	SCOPUS	-
Categorization using semi-supervised clustering	2008	Jianying Hu; Singh, M.; Mojsilovic, A.; , Categorization using semi-supervised clustering, Pattern Recognition, 2008. ICPR 2008. 19th International Conference on , vol., no., pp.1-4, 8-11 Dec. 2008	EC1	-	IEEE	-

Table A.1: Complete list returned by systematic mapping review.

IEEE Student Branch profile: Strength in numbers	2008	Causser, C.; , IEEE Student Branch profile: Strength in numbers, Potentials, IEEE , vol.27, no.6, pp.4-5, November-December 2008	EC1	-	IEEE	-
Lessons Learned from the SDSS Catalog Archive Server	2008	Thakar, A.R.; , Lessons Learned from the SDSS Catalog Archive Server, Computing in Science & Engineering , vol.10, no.6, pp.65-71, Nov.-Dec. 2008	EC1	-	IEEE	-
Relighting with real incident light source	2008	Yifan Chen; Xubo Yang; Shuangjiu Xiao; Xiaodong Ding; , Relighting with real incident light source, Mixed and Augmented Reality, 2008. ISMAR 2008. 7th IEEE/ACM International Symposium on , vol., no., pp.157-158, 15-18 Sept. 2008	EC1	-	IEEE	-
The changing landscape of multimedia SoC design	2008	Dutta, S.; , The changing landscape of multimedia SoC design, Consumer Electronics, 2008. ISCE 2008. IEEE International Symposium on , vol., no., pp.1, 14-16 April 2008	EC1	-	IEEE	-
The ENTHRONE 2 metadata management tool (MATool) (WISE 2008 MATool demonstration)	2008	Lugmayr, A.; , The ENTHRONE 2 metadata management tool (MATool) (WISE 2008 MATool demonstration), Computer Systems and Applications, 2008. AICCSA 2008. IEEE/ACS International Conference on , vol., no., pp.1013-1018, March 31 2008-April 4 2008	EC1	-	IEEE	-
Web-Based Intelligent CSCW Exploiting Context-Based Reasoning	2008	Sakurai, Y.; Gonzalez, A.J.; Nguyen, J.; Takada, K.; Uchida, K.; Tsuruta, S.; , Web-Based Intelligent CSCW Exploiting Context-Based Reasoning, Signal Image Technology and Internet Based Systems, 2008. SITIS '08. IEEE International Conference on , vol., no., pp.490-497, Nov. 30 2008-Dec. 3 2008	EC1	-	IEEE	-
Whither the CIO? Evolution from keeper of the infrastructure to firm innovator	2008	Fortino, A.; , Whither the CIO? Evolution from keeper of the infrastructure to firm innovator, Management of Engineering & Technology, 2008. PICMET 2008. Portland International Conference on , vol., no., pp.1878-1886, 27-31 July 2008	EC1	-	IEEE	-
Design and development of a terrestrial digital video broadcast demodulation core: An international collaborative effort	2007	Ashikhmin A., De Lind Van Wijngaarden A.J., Haibo Z., Hochwald B.M., Marzetta T.L., Purohit V., Qinghong C., Wilford P.A., Zhou S.-R., Zuniga M.A., Zuranski E.S.. Design and development of a terrestrial digital video broadcast demodulation core: An international collaborative effort. Bell Labs Technical Journal, 2007 pp 97 - 118.	EC1	-	SCOPUS	-
Art & complexity: An exploration of aesthetics	2007	Birkin G.. Art & complexity: An exploration of aesthetics. Creativity and Cognition 2007, CC2007 - Seeding Creativity: Tools, Media, and Environments, 2007 pp 278 - .	EC3	-	SCOPUS	-
Visualizing endangered indigenous languages of French Polynesia with LEXUS	2007	Cablitz G., Ringersma J., Kemps-Snijders M.. Visualizing endangered indigenous languages of French Polynesia with LEXUS. Proceedings of the International Conference on Information Visualisation, 2007 pp 409 - 414.	EC1	-	SCOPUS	-
The karst collaborative workspace for analyzing and annotating scientific datasets	2007	Collins L.M., Northup D.E., Martinez M.L.B., Van Reenen J., Baker M.A., Crowley C.R., Powell J.E., Freels-Stendel B., Heckethorn S.K., Park J.C.. The karst collaborative workspace for analyzing and annotating scientific datasets. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2007 pp 3 - 12.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

Development of fully automatic inspection systems for large underground concrete pipes partially filled with wastewater	2007	Elkmann N., Althoff H., Kutzner S., Stuerze T., Saenz J., Reimann B.. Development of fully automatic inspection systems for large underground concrete pipes partially filled with wastewater. Proceedings - IEEE International Conference on Robotics and Automation, 2007 pp 130 - 135.	EC1	-	SCOPUS	-
Finding the way: Improving access to the collections of the Royal Scottish Geographical Society	2007	Fenton C.. Finding the way: Improving access to the collections of the Royal Scottish Geographical Society. Program, 2007 pp 353 - 364.	EC1	-	SCOPUS	-
Agile-CAD for reverse engineering	2007	Ferreira R., Leal I., Alves N., Bartolo P.. Agile-CAD for reverse engineering. Proceedings of the 3rd International Conference on Advanced Research in Virtual and Rapid Prototyping: Virtual and Rapid Manufacturing Advanced Research Virtual and Rapid Prototyping, 2007 pp 257 - 261.	EC1	-	SCOPUS	-
An ASIC circuit for timing measurements with strip detectors, designed for the SiliPET project	2007	Gola A., Fiorini C., Di Domenico G., Zavattini G., Auricchio N.. An ASIC circuit for timing measurements with strip detectors, designed for the SiliPET project. IEEE Nuclear Science Symposium Conference Record, 2007 pp 370 - 374.	EC1	-	SCOPUS	-
The readout electronics and the DAQ system of the DRAGO Anger Camera	2007	Gola A., Fiorini C., Porro M., Zanchi M.. The readout electronics and the DAQ system of the DRAGO Anger Camera. IEEE Nuclear Science Symposium Conference Record, 2007 pp 1334 - 1337.	EC1	-	SCOPUS	-
Digital architectural reconstruction: New media technology and their use as educational tools in the Arabian Gulf	2007	Hawker R.W.. Digital architectural reconstruction: New media technology and their use as educational tools in the Arabian Gulf. WIT Transactions on the Built Environment, 2007 pp 587 - 595.	EC1	-	SCOPUS	-
The great Buddha project: Digitally archiving, restoring, and analyzing cultural heritage objects	2007	Ikeuchi K., Oishi T., Takamatsu J., Sagawa R., Nakazawa A., Kurazume R., Nishino K., Kamakura M., Okamoto Y.. The great Buddha project: Digitally archiving, restoring, and analyzing cultural heritage objects. International Journal of Computer Vision, 2007 pp 189 - 208.	EC1	-	SCOPUS	-
Library composition and adaptation using c++ concepts	2007	Jarvi J., Marcus M.A., Smith J.N.. Library composition and adaptation using c++ concepts. GPCE'07 - Proceedings of the Sixth International Conference on Generative Programming and Component Engineering, 2007 pp 73 - 82.	EC1	-	SCOPUS	-
Model-based design of an embedded vision application: A field report	2007	Kogler J., Hemetsberger H., Kubinger W., Borbely S.. Model-based design of an embedded vision application: A field report. Proceedings of the 4th IASTED International Conference on Signal Processing, Pattern Recognition, and Applications, SPPRA 2007, 2007 pp 233 - 238.	EC1	-	SCOPUS	-
Building an experimental infrastructure for B3G testing using an event-based distributed system	2007	Kormentzas G.. Building an experimental infrastructure for B3G testing using an event-based distributed system. 2007 16th IST Mobile and Wireless Communications Summit, 2007 pp - .	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

Design and implementation of gradient vector flow snake to detect a reference object in pelvic X-rays for preoperative total hip arthroplasty planning application	2007	Kristanto W., Van Ooijen P.M.A., The B., Duifhuis H., Mengko T.R., Oudkerk M.. Design and implementation of gradient vector flow snake to detect a reference object in pelvic X-rays for preoperative total hip arthroplasty planning application. Journal of Digital Imaging, 2007 pp 373 - 380.	EC1	-	SCOPUS	-
Quality assessment of the fire hazard forecast based on a fire potential index for the Mediterranean area by using a MSG/SEVIRI based fire detection system	2007	Laneve G., Cadau E.G.. Quality assessment of the fire hazard forecast based on a fire potential index for the Mediterranean area by using a MSG/SEVIRI based fire detection system. International Geoscience and Remote Sensing Symposium (IGARSS), 2007 pp 2447 - 2450.	EC1	-	SCOPUS	-
From endoscopic imaging and knowledge to semantic formal images	2007	Le Guillou C., Cauvin J.-M., Solaiman B., Robaszkiewicz M., Roux C.. From endoscopic imaging and knowledge to semantic formal images. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2007 pp 189 - 201.	EC1	-	SCOPUS	-
Lawrence Blaine is unwell: A web-based international 'community of practice' to engage nursing students in the planning and delivery of health care	2007	Lindsay B.. Lawrence Blaine is unwell: A web-based international 'community of practice' to engage nursing students in the planning and delivery of health care. Proceedings of the International Conference on e-Learning, ICEL, 2007 pp 299 - 306.	EC1	-	SCOPUS	-
Fuzzy colored timed petri nets for software project management	2007	Looney C.G., Dascalu S.. Fuzzy colored timed petri nets for software project management. 20th International Conference on Computer Applications in Industry and Engineering 2007, CAINE 2007, 2007 pp 168 - 173.	EC1	-	SCOPUS	-
Fusion of multispectral video sequences based on the trajectories of moving objects	2007	Morin F., Bilodeau G.-A.. Fusion of multispectral video sequences based on the trajectories of moving objects. ROSE 2007 - International Workshop on Robotic and Sensor Environments, Proceedings, 2007 pp 120 - 124.	EC1	-	SCOPUS	-
Improving cross-cultural communication through collaborative technologies	2007	O'Brien A.J., Alfano C., Magnusson E.. Improving cross-cultural communication through collaborative technologies. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2007 pp 125 - 131.	EC1	-	SCOPUS	-
Enhancing multimodal annotations with pen-based information	2007	Pimentel M.G., Goularte R., Cattelan R.G., Santos F.S., Teixeira C.. Enhancing multimodal annotations with pen-based information. Proceedings ISM Workshops 2007 9th IEEE International Symposium on Multimedia - Workshops, 2007 pp 207 - 212.	EC1	-	SCOPUS	-
Mastering DICOM with DVTK	2007	Potter G., Busbridge R., Toland M., Nagy P.. Mastering DICOM with DVTK. Journal of Digital Imaging, 2007 pp 47 - 62.	EC1	-	SCOPUS	-
Maximizing the adoption of fixed number portability within the EU: An empirical analysis	2007	Prezerakos G.N., Polykalas S.E.. Maximizing the adoption of fixed number portability within the EU: An empirical analysis. Telecommunications Policy, 2007 pp 179 - 196.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

Management competences, not tools and techniques: A grounded examination of software project management at WM-data	2007	Rose J., Pedersen K., Hosbond J.H., Kraemmergaard P.. Management competences, not tools and techniques: A grounded examination of software project management at WM-data. Information and Software Technology, 2007 pp 605 - 624.	EC1	-	SCOPUS	-
The ALOS Kyoto & carbon initiative	2007	Rosenqvist A., Shimada M., Milne A.K.. The ALOS Kyoto & carbon initiative. International Geoscience and Remote Sensing Symposium (IGARSS), 2007 pp 3614 - 3617.	EC1	-	SCOPUS	-
Algorithm visualization: A report on the state of the field	2007	Shaffer C.A., Cooper M., Edwards S.H.. Algorithm visualization: A report on the state of the field. SIGCSE 2007: 38th SIGCSE Technical Symposium on Computer Science Education, 2007 pp 150 - 154.	EC1	-	SCOPUS	-
An application of Google Earth for Forest Inventory in Alishan area	2007	Shih C.-H., Lau C.-C.. An application of Google Earth for Forest Inventory in Alishan area. 28th Asian Conference on Remote Sensing 2007, ACRS 2007, 2007 pp 1645 - 1652.	EC1	-	SCOPUS	-
A new approach to the computer support of strategic decision making in enterprises by means of a new class of understanding based management support systems	2007	Tadeusiewicz R., Ogiela M., Ogiela L.. A new approach to the computer support of strategic decision making in enterprises by means of a new class of understanding based management support systems. Proceedings - 6th International Conference on Computer Information Systems and Industrial Management Applications, CISIM 2007, 2007 pp 9 - 13.	EC1	-	SCOPUS	-
RAVE - An open, extensible, detector-independent toolkit for reconstruction of interaction vertices	2007	Waltenberger W., Moser F.. RAVE - An open, extensible, detector-independent toolkit for reconstruction of interaction vertices. IEEE Nuclear Science Symposium Conference Record, 2007 pp 104 - 109.	EC1	-	SCOPUS	-
Simulation of hounsfield units for a computed tomography scanner and different phantom inserts	2007	Wysocka-Rabin A., Qamhiyeh S., Jakel O.. Simulation of hounsfield units for a computed tomography scanner and different phantom inserts. EUROCON 2007 - The International Conference on Computer as a Tool, 2007 pp 2361 - 2366.	EC1	-	SCOPUS	-
Technology-enhanced language learning: A case study	2007	Yang S.C., Chen Y.-J.. Technology-enhanced language learning: A case study. Computers in Human Behavior, 2007 pp 860 - 879.	EC1	-	SCOPUS	-
3D part retrieval in product data management system	2007	You C.-F., Chen T.-P.. 3D part retrieval in product data management system. Computer-Aided Design and Applications, 2007 pp 117 - 125.	EC1	-	SCOPUS	-
Using hue, saturation, and value color space for hydraulic excavator idle time analysis	2007	Zou J., Kim H.. Using hue, saturation, and value color space for hydraulic excavator idle time analysis. Journal of Computing in Civil Engineering, 2007 pp 238 - 246.	EC1	-	SCOPUS	-
A General Architecture in Support of Personalized, Interactive Multimedia Services in the Mobile Broadcast Convergent Environment	2007	Wang Hui; Song Yali; Tang Xiaosheng; Zhang Ping; , A General Architecture in Support of Personalized, Interactive Multimedia Services in the Mobile Broadcast Convergent Environment, Testbeds and Research Infrastructure for the Development of Networks and Communities, 2007. TridentCom 2007. 3rd International Conference on , vol., no., pp.1-6, 21-23 May 2007	EC1	-	IEEE	-



Table A.1: Complete list returned by systematic mapping review.

A Management Platform for Multimedia Distribution in Country-wide Networks	2007	Uchoa, D.C.; Kulesza, R.; Matushima, R.; Kopp, S.; Bressan, G.; Silveira, R.M.; , A Management Platform for Multimedia Distribution in Country-wide Networks, Network Operations and Management Symposium, 2007. LANOMS 2007. Latin American , vol., no., pp.20-27, 10-12 Sept. 2007	EC1	-	IEEE	-
A Practical Color Transfer Algorithm for Image Sequences	2007	Yao-Hsien Huang; Chung-Hsin Liu; , A Practical Color Transfer Algorithm for Image Sequences, Intelligent Information Hiding and Multimedia Signal Processing, 2007. IHHMSP 2007. Third International Conference on , vol.1, no., pp.577-580, 26-28 Nov. 2007	EC1	-	IEEE	-
A Relevance Feedback Algorithm Based on SVM Model's Dynamic Adjusting for Image Retrieval	2007	Yihua Zhou; Weimin Shi; Lijuan Duan; Cuiying Niu; , A Relevance Feedback Algorithm Based on SVM Model's Dynamic Adjusting for Image Retrieval, Computational Intelligence and Security Workshops, 2007. CISW 2007. International Conference on , vol., no., pp.287-290, 15-19 Dec. 2007	EC1	-	IEEE	-
A study on development of real time monitoring system for field integrated management - overall automation of steel construction -	2007	Hyun Tae Ju; Chi Su Son; Kyung Hun Kim; Kyung Hwan Kim; Jae Jun Kim; , A study on development of real time monitoring system for field integrated management - overall automation of steel construction -, Control, Automation and Systems, 2007. ICCAS '07. International Conference on , vol., no., pp.1937-1941, 17-20 Oct. 2007	EC1	-	IEEE	-
Fault Management based on peer-to-peer paradigms; A case study report from the CELTIC project Madeira	2007	Leitner, M.; Leitner, P.; Zach, M.; Collins, S.; Fahy, C.; , Fault Management based on peer-to-peer paradigms; A case study report from the CELTIC project Madeira, Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on , vol., no., pp.697-700, May 21 2007-Yearly 25 2007	EC1	-	IEEE	-
Knowledge-Based Recognition of Utility Map Sub-Diagrams	2007	Hickinbotham, S.J.; Cohn, A.G.; , Knowledge-Based Recognition of Utility Map Sub-Diagrams, Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on , vol.1, no., pp.213-217, 23-26 Sept. 2007	EC1	-	IEEE	-
Management of QoS Metadata for Consuming Interactive Digital Media at Home	2007	Lugmayr, Artur; , Management of QoS Metadata for Consuming Interactive Digital Media at Home, Multimedia Workshops, 2007. ISMW '07. Ninth IEEE International Symposium on , vol., no., pp.221-226, 10-12 Dec. 2007	EC1	-	IEEE	-
openSourcePACS: An Extensible Infrastructure for Medical Image Management	2007	Bui, A.A.T.; Morioka, C.; Dionisio, J.D.N.; Johnson, D.B.; Sinha, U.; Ardekani, S.; Taira, R.K.; Aberle, D.R.; El-Saden, S.; Kangarloo, H.; , openSourcePACS: An Extensible Infrastructure for Medical Image Management, Information Technology in Biomedicine, IEEE Transactions on , vol.11, no.1, pp.94-109, Jan. 2007	EC1	-	IEEE	-
SISTER Service Information Scheduling and Transmission and Epg contRol - A new open source SW solution for local/regional operators	2007	Caravantes, J.R.L.; Fernandez, S.; Ceacero, C.; , SISTER Service Information Scheduling and Transmission and Epg contRol - A new open source SW solution for local/regional operators, Mobile and Wireless Communications Summit, 2007. 16th IST , vol., no., pp.1-5, 1-5 July 2007	EC1	-	IEEE	-

Table A.1: Complete list returned by systematic mapping review.

Using A Convective Weather Forecast Product to Predict Weather Impact on Air Traffic: Methodology and Comparison with Actual Data	2007	Klein, A.; Kavoussi, S.; Hickman, D.; Simenauer, D.; Phaneuf, M.; MacPhail, T.; , Using A Convective Weather Forecast Product to Predict Weather Impact on Air Traffic: Methodology and Comparison with Actual Data, Integrated Communications, Navigation and Surveillance Conference, 2007. ICNS '07 , vol., no., pp.1-10, April 30 2007-May 3 2007	EC1	-	IEEE	-
Construction workspace planning: Assignment and analysis utilizing 4D visualization technologies	2006	Dawood N., Mallasi Z.. Construction workspace planning: Assignment and analysis utilizing 4D visualization technologies. Computer-Aided Civil and Infrastructure Engineering, 2006 pp 498 - 513.	EC1	-	SCOPUS	-
WORLDMAPPER: The world as you've never seen it before	2006	Dorling D., Barford A., Newman M.. WORLDMAPPER: The world as you've never seen it before. IEEE Transactions on Visualization and Computer Graphics, 2006 pp 757 - 764.	EC1	-	SCOPUS	-
Single organ segmentation filters for multiple organ segmentation.	2006	Furst J.D., Susomboom R., Raicu D.S.. Single organ segmentation filters for multiple organ segmentation.. Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference, 2006 pp 3033 - 3036.	EC1	-	SCOPUS	-
Video traffic management in HSDPA via GEO satellite	2006	Giambene G., Giannetti S., Niebla C.P., Ries M.. Video traffic management in HSDPA via GEO satellite. 2006 International Workshop on Satellite and Space Communications, IWSSC, 2006 pp 188 - 192.	EC1	-	SCOPUS	-
Evaluating evaluation: Introducing a research project on the impact of improve your library: A self-evaluation process for school libraries	2006	Gildersleeves L.. Evaluating evaluation: Introducing a research project on the impact of improve your library: A self-evaluation process for school libraries. Aslib Proceedings: New Information Perspectives, 2006 pp 73 - 88.	EC1	-	SCOPUS	-
DARPA autonomous airborne refueling demonstration program with initial results	2006	Hansen J., Romrell G., Nabaa N., Andersen R., Myers L., McCormick J.. DARPA autonomous airborne refueling demonstration program with initial results. Proceedings of the Institute of Navigation - 19th International Technical Meeting of the Satellite Division, ION GNSS 2006, 2006 pp 674 - 685.	EC1	-	SCOPUS	-
Vision guided manipulator for optimal dynamic performance	2006	Jamaluddin M.H., Said M.A., Sulaiman M., Horng C.S.. Vision guided manipulator for optimal dynamic performance. SCORED 2006 - Proceedings of 2006 4th Student Conference on Research and Development Towards Enhancing Research Excellence in the Region, 2006 pp 147 - 151.	EC1	-	SCOPUS	-
Synchronization strategies for spatial information organization	2006	Kukulenz D., Kasper J.. Synchronization strategies for spatial information organization. Proceedings of the International Conference on Information Visualisation, 2006 pp 174 - 180.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

The global lambda visualization facility: An international ultra-high-definition wide-area visualization collaboration	2006	Leigh J., Renambot L., Johnson A., Jeong B., Jagodic R., Schwarz N., Svistula D., Singh R., Aguilera J., Wang X., Vishwanath V., Lopez B., Sandin D., Peterka T., Girado J., Kooima R., Ge J., Long L., Verlo A., DeFanti T.A., Brown M., Cox D., Patterson R., Dorn P., Wefel P., Levy S., Talandis J., Reitzer J., Prudhomme T., Coffin T., Davis B., Wielinga P., Stolk B., Bum Koo G., Kim J., Han S., Kim J., Corrie B., Zimmerman T., Boulanger P., Garcia M.. The global lambda visualization facility: An international ultra-high-definition wide-area visualization collaboration. Future Generation Computer Systems, 2006 pp 964 - 971.	EC1	-	SCOPUS	-
A benchmark approach for compilers in reconfigurable hardware	2006	Lopes J.J., Silva J.L.E., Marques E., Cardoso J.M.P.. A benchmark approach for compilers in reconfigurable hardware. Proceedings - The 6th IEEE International Workshop on System on Chip for Real Time Applications, IWSOC 2006, 2006 pp 120 - 124.	EC1	-	SCOPUS	-
Operational exploitation of QuickBird imagery for high accuracy airport mapping	2006	Low C., Bannerman K., Roberston B.C., Brunke S., Martin S.. Operational exploitation of QuickBird imagery for high accuracy airport mapping. International Geoscience and Remote Sensing Symposium (IGARSS), 2006 pp 4209 - 4212.	EC1	-	SCOPUS	-
Metadata handling: A video perspective	2006	Madhwacharyula C.L., Davis M., Mulhem P., Kankanhalli M.S.. Metadata handling: A video perspective. ACM Transactions on Multimedia Computing, Communications and Applications, 2006 pp 358 - 388.	EC1	-	SCOPUS	-
Development and design of a multimedia tool for technology-transfer in industrial processes	2006	Marino P., Dominguez M.A., Otero S., Merino M.. Development and design of a multimedia tool for technology-transfer in industrial processes. EISTA 2006 - 4th Int. Conf. on Education and Information Systems: Technologies and Applications, Jointly with SOIC 2006 - 2nd Int. Conf. on SOIC and PISTA 2006 - 4th Int. Conf. on PISTA, Proceedings, 2006 pp 118 - 122.	EC1	-	SCOPUS	-
A multi-year data set of cloud properties derived for CERES from Aqua, Terra, and TRMM	2006	Minnis P., Sun-Mack S., Trepte Q.Z., Chen Y., Brown R.R., Gibson S., Heck P.W., Dong X., Xi B.. A multi-year data set of cloud properties derived for CERES from Aqua, Terra, and TRMM. International Geoscience and Remote Sensing Symposium (IGARSS), 2006 pp 1780 - 1783.	EC1	-	SCOPUS	-
Classification for the ripeness of papayas using artificial neural network (ANN) and threshold rule	2006	Saad H., Hussain A.. Classification for the ripeness of papayas using artificial neural network (ANN) and threshold rule. SCOREd 2006 - Proceedings of 2006 4th Student Conference on Research and Development Towards Enhancing Research Excellence in the Region, 2006 pp 132 - 136.	EC1	-	SCOPUS	-
Automated acquisition planning for commercial satellite imagery	2006	Seeker J., Rowe J., Robson M., Vachon P.W.. Automated acquisition planning for commercial satellite imagery. International Geoscience and Remote Sensing Symposium (IGARSS), 2006 pp 3279 - 3282.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

Fraction images derived from terra MODIS data for mapping burned area in Acre State, Brazilian Amazonia	2006	Shimabukuro Y.E., Duarte V., Arai E., De Freitas R.M., De Morrison Valeriano D., De Sensoriamento Remoto D., Brown I.F., De Los Rios Maldonado M.. Fraction images derived from terra MODIS data for mapping burned area in Acre State, Brazilian Amazonia. International Geoscience and Remote Sensing Symposium (IGARSS), 2006 pp 4161 - 4164.	EC1	-	SCOPUS	-
3D navigable interface for interactive movie Gormenghast Explore	2006	Sussner J., Lohse L., Thomas M., Garcia G., Alonso I., Munoz A.. 3D navigable interface for interactive movie Gormenghast Explore. Proceedings - Second International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution, AXMEDIS 2006, 2006 pp 242 - 247.	EC1	-	SCOPUS	-
A tool for concurrent image development	2006	Wong J., Capretz M.A.M.. A tool for concurrent image development. WSEAS Transactions on Computers, 2006 pp 2364 - 2371.	OK	EC4	SCOPUS	Image
A product data dependencies network to support conflict resolution in design processes	2006	Ouertanf, M.Z.; Grebici, K.; Gzara-Yesilbas, L.; Blanco, E.; Rieu, D.; , A product data dependencies network to support conflict resolution in design processes, Computational Engineering in Systems Applications, IMACS Multiconference on , vol.2, no., pp.1189-1196, 4-6 Oct. 2006	EC1	-	IEEE	-
A product data dependencies network to support conflict resolution in design processes	2006	Ouertanf, M.Z.; Grebici, K.; Gzara-Yesilbas, L.; Blanco, E.; Rieu, D.; , A product data dependencies network to support conflict resolution in design processes, Computational Engineering in Systems Applications, IMACS Multiconference on , vol., no., pp.1189-1196, 4-6 Oct. 2006	EC1	-	IEEE	-
Animated Visualization of Software History using Evolution Storyboards	2006	Dirk Beyer; Ahmed E. Hassan; , Animated Visualization of Software History using Evolution Storyboards, Reverse Engineering, 2006. WCRE '06. 13th Working Conference on , vol., no., pp.199-210, Oct. 2006	EC1	-	IEEE	-
Appropriateness of e-learning resources for the development of transversal skills in the new European Higher Education Area	2006	Vicent, L.; Avila, X.; Riera, J.; Badia, D.; Anguera, J.; Montero, J.A.; , Appropriateness of e-learning resources for the development of transversal skills in the new European Higher Education Area, Frontiers in Education Conference, 36th Annual , vol., no., pp.6-11, 27-31 Oct. 2006	EC1	-	IEEE	-
Challenges in the Adoption of Medical Information Systems	2006	Maass, M.; Eriksson, O.; , Challenges in the Adoption of Medical Information Systems, System Sciences, 2006. HICSS '06. Proceedings of the 39th Annual Hawaii International Conference on , vol.5, no., pp. 95b, 04-07 Jan. 2006	EC1	-	IEEE	-
Distributed Scheduling for WIMAX Mesh Network	2006	Makarevitch, B.; , Distributed Scheduling for WIMAX Mesh Network, Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium on , vol., no., pp.1-5, 11-14 Sept. 2006	EC1	-	IEEE	-
Focusing ISAR images using the AJTF optimized with the GA and the PSO algorithm-comparison and results	2006	Brinkman, W.; Thayananthan Thayaparan; , Focusing ISAR images using the AJTF optimized with the GA and the PSO algorithm-comparison and results, Radar, 2006 IEEE Conference on , vol., no., pp. 8 pp., 24-27 April 2006	EC1	-	IEEE	-

Table A.1: Complete list returned by systematic mapping review.

Multi-dimensional Dependency and Conflict Resolution for Self-adaptable Context-aware Systems	2006	Preuveneers, D.; Berbers, Y.; , Multi-dimensional Dependency and Conflict Resolution for Self-adaptable Context-aware Systems, Autonomic and Autonomous Systems, 2006. ICAS '06. 2006 International Conference on , vol., no., pp.36, 16-18 July 2006	EC1	-	IEEE	-
Single Organ Segmentation Filters for Multiple Organ Segmentation	2006	Furst, J.D.; Susomboom, R.; Raicu, D.S.; , Single Organ Segmentation Filters for Multiple Organ Segmentation, Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE , vol., no., pp.3033-3036, Aug. 30 2006-Sept. 3 2006	EC1	-	IEEE	-
Towards an Efficient Integration, Structure and Exploration of Landscape Architecture Project Information	2006	Favetta, F.; Laurini, R.; , Towards an Efficient Integration, Structure and Exploration of Landscape Architecture Project Information, Multimedia and Expo, 2006 IEEE International Conference on , vol., no., pp.397-400, 9-12 July 2006	EC1	-	IEEE	-
Volume estimation from uncalibrated views applied to wound measurement	2005	Albouy B., Treuillet S., Lucas Y., Pichaud J.C.. Volume estimation from uncalibrated views applied to wound measurement. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2005 pp 945 - 952.	EC1	-	SCOPUS	-
Eye tracking in coloured image scenes represented by ambisonic fields of musical instrument sounds	2005	Bologna G., Vinckenbosch M.. Eye tracking in coloured image scenes represented by ambisonic fields of musical instrument sounds. Lecture Notes in Computer Science, 2005 pp 327 - 337.	EC1	-	SCOPUS	-
Comparing faculty information seeking in teaching and research: Implications for the design of digital libraries	2005	Borgman C.L., Smart L.J., Millwood K.A., Finley J.R., Champeny L., Gilliland A.J., Leazer G.H.. Comparing faculty information seeking in teaching and research: Implications for the design of digital libraries. Journal of the American Society for Information Science and Technology, 2005 pp 636 - 657.	EC1	-	SCOPUS	-
Managing algorithmic skeleton nesting requirements in realistic image processing applications: The case of the SKiPPER-II Parallel Programming EnviRonment's operating model	2005	Coudarcher R., Duculty F., Serot J., Jurie F., Derutin J.-P., Dhôme M.. Managing algorithmic skeleton nesting requirements in realistic image processing applications: The case of the SKiPPER-II Parallel Programming EnviRonment's operating model. Eurasip Journal on Applied Signal Processing, 2005 pp 1005 - 1023.	EC1	-	SCOPUS	-
Influences of image disturbances on 2D face recognition	2005	Daum H.. Influences of image disturbances on 2D face recognition. Lecture Notes in Computer Science, 2005 pp 900 - 908.	EC1	-	SCOPUS	-
Global seamless network demonstrator: A comprehensive ASON/GMPLS testbed	2005	Foisel H.-M., Gerlach C., Gladisch A., Szuppa S., Weber A.. Global seamless network demonstrator: A comprehensive ASON/GMPLS testbed. IEEE Communications Magazine, 2005 pp - .	EC1	-	SCOPUS	-
Using photography as a metaphor for teaching image synthesis	2005	Geigel J., Schaller N.C.. Using photography as a metaphor for teaching image synthesis. Computers and Graphics (Pergamon), 2005 pp 257 - 265.	EC1	-	SCOPUS	-
Reconstructing camera projection matrices from multiple pairwise overlapping views	2005	Goldberger J.. Reconstructing camera projection matrices from multiple pairwise overlapping views. Computer Vision and Image Understanding, 2005 pp 283 - 296.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

Painless project management with FogBugz	2005	Gunderloy M.. Painless project management with FogBugz. Painless Project Management with FogBugz, 2005 pp 1 - 184.	EC1	-	SCOPUS	-
Color quality analysis of a system for digital distribution and projection of cinema commercials	2005	Hardeberg J.Y., Farub I., Stjernvang G.. Color quality analysis of a system for digital distribution and projection of cinema commercials. SMPTE Motion Imaging Journal, 2005 pp 146 - 151.	EC1	-	SCOPUS	-
Capturing content for virtual museums: From pieces to exhibits	2005	Hemminger B., Bolas G., Schiff D.. Capturing content for virtual museums: From pieces to exhibits. Journal of Digital Information, 2005 pp - .	EC1	-	SCOPUS	-
Incorporating 3D virtual anatomy into the medical curriculum	2005	Imielinska C., Molholt P.. Incorporating 3D virtual anatomy into the medical curriculum. Communications of the ACM, 2005 pp 49 - 54.	EC1	-	SCOPUS	-
Pattern recognition techniques for the emerging field of bioinformatics: A review	2005	Liew A.W.-C., Yan H., Yang M.. Pattern recognition techniques for the emerging field of bioinformatics: A review. Pattern Recognition, 2005 pp 2055 - 2073.	EC1	-	SCOPUS	-
Project management in multi-disciplinary collaborative research	2005	Lloyd S., Simpson A.. Project management in multi-disciplinary collaborative research. IEEE International Professional Communication Conference, 2005 pp 602 - 611.	OK	EC1	SCOPUS	-
Too much or too little: Visual considerations of public engagement tools in environment impact assessments	2005	Mak A.S.-H., Lai P.-C., Kwong R.K.-H., Leung S.T.-S.. Too much or too little: Visual considerations of public engagement tools in environment impact assessments. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2005 pp 189 - 202.	EC1	-	SCOPUS	-
Researching ERP adoption: An internet-based grounded theory approach	2005	Oliver D., Whymark G., Romm C.. Researching ERP adoption: An internet-based grounded theory approach. Online Information Review, 2005 pp 585 - 603.	EC1	-	SCOPUS	-
Active appearance-based robot localization using stereo vision	2005	Porta J.M., Verbeek J.J., Krose B.J.A.. Active appearance-based robot localization using stereo vision. Autonomous Robots, 2005 pp 59 - 80.	EC1	-	SCOPUS	-
Designing image processing software for those without a computer science degree	2005	Reineke N.. Designing image processing software for those without a computer science degree. Scientific Computing and Instrumentation, 2005 pp 27 - 28.	EC1	-	SCOPUS	-
Image interpolation for virtual sports scenarios	2005	Rodriguez T., Reid I., Horaud R., Dalal N., Goetz M.. Image interpolation for virtual sports scenarios. Machine Vision and Applications, 2005 pp 236 - 245.	EC1	-	SCOPUS	-
Photorealistic 3D reconstruction from handheld cameras	2005	Rodriguez T., Sturm P., Gargallo P., Guilbert N., Heyden A., Jauregizar F., Menendez J.M., Ronda J.I.. Photorealistic 3D reconstruction from handheld cameras. Machine Vision and Applications, 2005 pp 246 - 257.	EC1	-	SCOPUS	-
Run-time reconfigurable hardware blocks for multimedia applications	2005	Sakr N., Groza V.. Run-time reconfigurable hardware blocks for multimedia applications. Canadian Conference on Electrical and Computer Engineering, 2005 pp 1996 - 1999.	EC1	-	SCOPUS	-
Multimodal video indexing: A review of the state-of-the-art	2005	Snoek C.G.M., Worring M.. Multimodal video indexing: A review of the state-of-the-art. Multimedia Tools and Applications, 2005 pp 5 - 35.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

Revision control practices applied to computer configurations	2005	Sprague R.. Revision control practices applied to computer configurations. Proceedings ACM SIGUCCS User Services Conference, 2005 pp 356 - 359.	EC1	-	SCOPUS	-
Bayesian method for motion segmentation and tracking in compressed videos	2005	Treetasanatavorn S., Rauschenbach U., Heuer J., Kaup A.. Bayesian method for motion segmentation and tracking in compressed videos. Lecture Notes in Computer Science, 2005 pp 277 - 284.	EC1	-	SCOPUS	-
E-learning: What the literature tells us about distance education: An overview	2005	Williams P., Nicholas D., Gunter B.. E-learning: What the literature tells us about distance education: An overview. Aslib Proceedings: New Information Perspectives, 2005 pp 109 - 122.	EC1	-	SCOPUS	-
Open source software for medical image processing and visualization	2005	Yoo T.S., Ackerman M.J.. Open source software for medical image processing and visualization. Communications of the ACM, 2005 pp 55 - 59.	EC1	-	SCOPUS	-
An evaluation system for string extraction in the airline coupon project	2005	Yan Heping; Zhiyan Wang; Sen Guo; , An evaluation system for string extraction in the airline coupon project, Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on , vol., no., pp. 930- 934 Vol. 2, 29 Aug.-1 Sept. 2005	EC1	-	IEEE	-
Enriching multimedia content description for broadcast environments: from a unified metadata model to a new generation of authoring tool	2005	Rousseau, B.; Jouve, W.; Berti-Equille, L.; , Enriching multimedia content description for broadcast environments: from a unified metadata model to a new generation of authoring tool, Multimedia, Seventh IEEE International Symposium on , vol., no., pp. 8 pp., 12-14 Dec. 2005	EC1	-	IEEE	-
Introducing agile development (XP) into a corporate Webmaster environment - an experience report	2005	Ganis, M.; Leip, D.; Grossman, F.; Bergin, J.; , Introducing agile development (XP) into a corporate Webmaster environment - an experience report, Agile Conference, 2005. Proceedings , vol., no., pp. 145- 152, 24-29 July 2005	EC1	-	IEEE	-
Workflow-based Remote-Sensing Image Processing Application in ImageGrid	2005	Ran Zheng; Hai Jin; Qin Zhang; Ying Li; , Workflow-based Remote-Sensing Image Processing Application in ImageGrid, Parallel and Distributed Computing, Applications and Technologies, 2005. PDCAT 2005. Sixth International Conference on , vol., no., pp. 390- 394, 05-08 Dec. 2005	EC1	-	IEEE	-
Codesign methodology for computer vision applications	2004	Albaladejo J., De Andres D., Lemus L., Salvi J.. Codesign methodology for computer vision applications. Microprocessors and Microsystems, 2004 pp 303 - 316.	EC1	-	SCOPUS	-
Grid databases for shared image analysis in the mammoGrid project	2004	Amendolia S.R., Estrella F., Hauer T., Manset D., McClatchey R., Odeh M., Reading T., Rogulin D., Schottlander D., Solomonides T.. Grid databases for shared image analysis in the mammoGrid project. Proceedings of the International Database Engineering and Applications Symposium, IDEAS, 2004 pp 302 - 311.	EC1	-	SCOPUS	-
Managing a portfolio of overlay paths	2004	Antonova D., Krishnamurthy A., Ma Z., Sundaram R.. Managing a portfolio of overlay paths. Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video, 2004 pp 30 - 35.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

Spin images for retrieval of 3D objects by local and global similarity	2004	Assfalg J., Del Bimbo A., Pala P.. Spin images for retrieval of 3D objects by local and global similarity. Proceedings - International Conference on Pattern Recognition, 2004 pp 906 - 909.	EC1	-	SCOPUS	-
MatDL: Integrating digital libraries into scientific practice	2004	Bartolo L.M., Lowe C.S., Feng L.Z., Patten B.. MatDL: Integrating digital libraries into scientific practice. Journal of Digital Information, 2004 pp - .	EC1	-	SCOPUS	-
Get more work out of your day	2004	Bass S.. Get more work out of your day. PC World (San Francisco, CA), 2004 pp 55 - .	EC1	-	SCOPUS	-
RAW: Conveying minimally-mediated impressions of everyday life with an audio-photographic tool	2004	Bitton J., Agamanolis S., Karau M.. RAW: Conveying minimally-mediated impressions of everyday life with an audio-photographic tool. Conference on Human Factors in Computing Systems - Proceedings, 2004 pp 495 - 502.	EC1	-	SCOPUS	-
Lighting up storage	2004	Carr J.. Lighting up storage. Network Magazine, 2004 pp 72 - 74.	EC1	-	SCOPUS	-
Change detection and analysis of housebreaking in 2008 Beijing olympic main venue using airborne remote sensing photos	2004	Chen X., Dai Q., Feng C., Ma J.. Change detection and analysis of housebreaking in 2008 Beijing olympic main venue using airborne remote sensing photos. International Geoscience and Remote Sensing Symposium (IGARSS), 2004 pp 3884 - 3887.	EC1	-	SCOPUS	-
Texture classification using Kernel independent component analysis	2004	Cheng J., Liu Q., Lu H., Chen Y.-W.. Texture classification using Kernel independent component analysis. Proceedings - International Conference on Pattern Recognition, 2004 pp 620 - 623.	EC1	-	SCOPUS	-
Did the great masters use optical projections while painting? Perspective comparison of paintings and photographs of Renaissance chandeliers	2004	Criminisi A., Stork D.G.. Did the great masters use optical projections while painting? Perspective comparison of paintings and photographs of Renaissance chandeliers. Proceedings - International Conference on Pattern Recognition, 2004 pp 645 - 648.	EC1	-	SCOPUS	-
Temporal soil moisture estimates from Radarsat-1 and Envisat ASAR for flood forecasting	2004	Deschamps A., Pultz T.J., Pietroniro A., Best K.. Temporal soil moisture estimates from Radarsat-1 and Envisat ASAR for flood forecasting. International Geoscience and Remote Sensing Symposium (IGARSS), 2004 pp 2119 - 2122.	EC1	-	SCOPUS	-
A bayesian framework for robust human detection and occlusion handling using human shape model	2004	Eng H.-L., Wang J., Kam A.H., Yau W.-Y.. A bayesian framework for robust human detection and occlusion handling using human shape model. Proceedings - International Conference on Pattern Recognition, 2004 pp 257 - 260.	EC1	-	SCOPUS	-
Landslide risk analysis by means of remote sensing techniques: Results from the ESA/SLAM project	2004	Farina P., Moretti S., Colombo D., Fumagalli A., Manunta P.. Landslide risk analysis by means of remote sensing techniques: Results from the ESA/SLAM project. International Geoscience and Remote Sensing Symposium (IGARSS), 2004 pp 62 - 65.	EC1	-	SCOPUS	-
Detection of optic disc in retinal images by means of a geometrical model of vessel structure	2004	Foracchia M., Grisan E., Ruggeri A.. Detection of optic disc in retinal images by means of a geometrical model of vessel structure. IEEE Transactions on Medical Imaging, 2004 pp 1189 - 1195.	EC1	-	SCOPUS	-
Edwards plateau: Analysis of land cover trends	2004	Friesen B.A., Hester D.J., Casey K.A.. Edwards plateau: Analysis of land cover trends. International Geoscience and Remote Sensing Symposium (IGARSS), 2004 pp 2639 - 2642.	EC1	-	SCOPUS	-



Table A.1: Complete list returned by systematic mapping review.

Locating text in historical collection manuscripts	2004	Gatos B., Pratikakis I., Perantonis S.J.. Locating text in historical collection manuscripts. Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science), 2004 pp 476 - 485.	EC1	-	SCOPUS	-
Project scheduling with irregular costs: Complexity, approximability, and algorithms	2004	Grigoriev A., Woeginger G.J.. Project scheduling with irregular costs: Complexity, approximability, and algorithms. Acta Informatica, 2004 pp 83 - 97.	EC1	-	SCOPUS	-
Project-based, asynchronous collaborative learning	2004	Hafner W., Ellis T.J.. Project-based, asynchronous collaborative learning. Proceedings of the Hawaii International Conference on System Sciences, 2004 pp 197 - 206.	EC1	-	SCOPUS	-
Application of soft computing to automatic music information retrieval	2004	Kostek B.. Application of soft computing to automatic music information retrieval. Journal of the American Society for Information Science and Technology, 2004 pp 1108 - 1116.	EC1	-	SCOPUS	-
Multi-level anchorperson detection using multimodal association	2004	Lan D.-J., Ma Y.-F., Zhang H.-J.. Multi-level anchorperson detection using multimodal association. Proceedings - International Conference on Pattern Recognition, 2004 pp 890 - 893.	EC1	-	SCOPUS	-
Fusing a laser range finder and a stereo vision system to detect obstacles in 3D	2004	Romero L., Nunez A., Bravo S., Gamboa L.E.. Fusing a laser range finder and a stereo vision system to detect obstacles in 3D. Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science), 2004 pp 555 - 561.	EC1	-	SCOPUS	-
I/O brush: Drawing with everyday objects as ink	2004	Ryokai K., Marti S., Ishii H.. I/O brush: Drawing with everyday objects as ink. Conference on Human Factors in Computing Systems - Proceedings, 2004 pp 303 - 310.	EC1	-	SCOPUS	-
Virtual Building for Construction Projects	2004	Sheppard L.M., Potel M.. Virtual Building for Construction Projects. IEEE Computer Graphics and Applications, 2004 pp 6 - 12.	EC1	-	IEEE, SCOPUS	-
Deforestation detection in brazilian amazon region in a near real time using terra modis daily data	2004	Shimabukuro Y.E., Duarte V., Anderson L.O., Arai E., Valeriano D.M., Santo F.D.B.E., Aulicino L.C.M.. Deforestation detection in brazilian amazon region in a near real time using terra modis daily data. International Geoscience and Remote Sensing Symposium (IGARSS), 2004 pp 3405 - 3408.	EC1	-	SCOPUS	-
Liquid edition 5.5	2004	Singer D.. Liquid edition 5.5. Computer Graphics World, 2004 pp 42 - .	EC1	-	SCOPUS	-
Xbox security issues and forensic recovery methodology (utilising linux)	2004	Vaughan C.. Xbox security issues and forensic recovery methodology (utilising linux). Digital Investigation, 2004 pp 165 - 172.	EC1	-	SCOPUS	-
Toys today; engineers tomorrow?	2004	Weber A.. Toys today; engineers tomorrow?. Assembly, 2004 pp 72 - 74.	EC1	-	SCOPUS	-
ASTER - A geological mapping tool for Canada's north. Case study: The Belcher Islands, Hudson Bay, Nunavut, Canada	2004	Wickert L.M., Budkewitsch P.. ASTER - A geological mapping tool for Canada's north. Case study: The Belcher Islands, Hudson Bay, Nunavut, Canada. International Geoscience and Remote Sensing Symposium (IGARSS), 2004 pp 1300 - 1303.	EC1	-	SCOPUS	-
Rendering complexity in computer-generated pen-and-ink illustrations	2004	Wilson B., Ma K.-L.. Rendering complexity in computer-generated pen-and-ink illustrations. NPAR Symposium on Non-Photorealistic Animation and Rendering, 2004 pp 103 - 111.	EC1	-	SCOPUS	-
A novel approach to detecting adult images	2004	Yang J., Fu Z., Tan T., Hu W.. A novel approach to detecting adult images. Proceedings - International Conference on Pattern Recognition, 2004 pp 479 - 482.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

GenExplore: Interactive exploration of gene interactions from microarray data	2004	Ye Y., Wu X., Subramanian K.R., Zhang L.. GenExplore: Interactive exploration of gene interactions from microarray data. Proceedings - International Conference on Data Engineering, 2004 pp 860 - .	EC1	-	SCOPUS	-
A novel approach to design classifiers using genetic programming	2004	Muni, D.P.; Pal, N.R.; Das, J.; , A novel approach to design classifiers using genetic programming, Evolutionary Computation, IEEE Transactions on , vol.8, no.2, pp. 183-196, April 2004	EC1	-	IEEE	-
Creating cyberworlds: experiences in computer science education	2004	Gutierrez, M.; Thalmann, D.; Vexo, F.; , Creating cyberworlds: experiences in computer science education, Cyberworlds, 2004 International Conference on , vol., no., pp. 401- 408, 18-20 Nov. 2004	EC1	-	IEEE	-
Modelling intelligent agents for organisational memories	2003	Arenas A.E., Barrera-Sanabria G.. Modelling intelligent agents for organisational memories. Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science), 2003 pp 430 - 437.	EC1	-	SCOPUS	-
Contextual Analysis of Multisource Raster Data by a Region-based Segmentation Tool in a Voronoi Structure	2003	Chakroun H., Benie G.B.. Contextual Analysis of Multisource Raster Data by a Region-based Segmentation Tool in a Voronoi Structure. International Geoscience and Remote Sensing Symposium (IGARSS), 2003 pp 1788 - 1792.	EC1	-	SCOPUS	-
Accelerated image processing on FPGAs	2003	Draper B.A., Ross Beveridge J., Willem Bohm A.P., Ross C., Chawathe M.. Accelerated image processing on FPGAs. IEEE Transactions on Image Processing, 2003 pp 1543 - 1551.	EC1	-	SCOPUS	-
A 3D-TV system based on video plus depth information	2003	Fehn C.. A 3D-TV system based on video plus depth information. Conference Record of the Asilomar Conference on Signals, Systems and Computers, 2003 pp 1529 - 1533.	EC1	-	SCOPUS	-
Panoptes: Scalable low-power video sensor networking technologies	2003	Feng W.-C., Code B., Kaiser E., Shea M., Feng W.-C.. Panoptes: Scalable low-power video sensor networking technologies. Proceedings of the ACM International Multimedia Conference and Exhibition, 2003 pp 90 - 91.	EC1	-	SCOPUS	-
ELFNI (Electronic Libraries for Northern Ireland project): An overview	2003	Frawley R.. ELFNI (Electronic Libraries for Northern Ireland project): An overview. Program, 2003 pp 94 - 102.	EC1	-	SCOPUS	-
Macintosh OS X - A Smooth Migration	2003	Hanselman S.E., Pegah M.. Macintosh OS X - A Smooth Migration. 31st Annual ACM SIGUCCS Fall Conference (SIGUCCS Conference Proceedings, 2003 pp 129 - 134.	EC1	-	SCOPUS	-
Needs and Trends of IT-Based Construction Field Data Collection	2003	Hwang S., Trupp T., Liu L.. Needs and Trends of IT-Based Construction Field Data Collection. Towards a Vision for Information Technology in Civil Engineering, 2003 pp 77 - 85.	EC1	-	SCOPUS	-
Four-primary-color LCD for natural vision	2003	Komura S., Hiyama I., Ohyama N.. Four-primary-color LCD for natural vision. Information Display, 2003 pp 18 - 21.	EC1	-	SCOPUS	-
Coordinating dependencies in complex system development projects	2003	Lillieskold J.. Coordinating dependencies in complex system development projects. IEEE International Engineering Management Conference, 2003 pp 400 - 404.	EC1	-	SCOPUS	-
Embedded GNU/Linux - Drawing the big-picture	2003	Mc Guire N.. Embedded GNU/Linux - Drawing the big-picture. Proceedings of the IEEE International Conference on Industrial Technology, 2003 pp 1237 - 1242.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

Computer aided diagnosis for breast masses detection on a telemammography system	2003	Mendez A.J., Souto M., Tahoces P.G., Vidal J.J.. Computer aided diagnosis for breast masses detection on a telemammography system. Computerized Medical Imaging and Graphics, 2003 pp 497 - 502.	EC1	-	SCOPUS	-
Using High-Capacity Data Networks and Uncompressed Video Transmissions for Distributed Television Productions in Real-Time	2003	Naegele-Jackson S., Holleczeck P., Metz A.. Using High-Capacity Data Networks and Uncompressed Video Transmissions for Distributed Television Productions in Real-Time. Proceedings of the International Conference on Communications in Computing, 2003 pp 126 - 129.	EC1	-	SCOPUS	-
An Approach for Designing Ubiquitous Web Applications: A Case Study	2003	Perrone V., Paolini P.. An Approach for Designing Ubiquitous Web Applications: A Case Study. Proceedings of the Second IASTED International Conference on Communications, Internet, and Information Technology, 2003 pp 348 - 354.	EC1	-	SCOPUS	-
Pixel classification through divergence-based integration of texture methods with conflict resolution	2003	Puig D., Garcia M.A.. Pixel classification through divergence-based integration of texture methods with conflict resolution. IEEE International Conference on Image Processing, 2003 pp 1037 - 1040.	EC1	-	SCOPUS	-
MINERVA: An INSAR monitoring system for volcanic hazard	2003	Usai S., Sansosti E., Tampellini L., Borgstrom S., Ricciardi G., Spaans J., Pepe A., Guarino S., Maddalena V., Van Persie M., Berardino P., Lanari R., Fornaro G., Seifert F.M.. MINERVA: An INSAR monitoring system for volcanic hazard. International Geoscience and Remote Sensing Symposium (IGARSS), 2003 pp 2433 - 2435.	EC1	-	SCOPUS	-
Boosting first-pass yield	2003	Wagner S., Clark D.. Boosting first-pass yield. Assembly, 2003 pp 38 - 45.	EC1	-	SCOPUS	-
A synchronous groupware and some scenarios for conducting a software engineering lab with distributed teams	2003	Werner S., Hunger A., Schwarz F., Schutz C., Jung M.. A synchronous groupware and some scenarios for conducting a software engineering lab with distributed teams. Proceedings of the IASTED International Conference on Computers and Advanced Technology in Education, 2003 pp 636 - 641.	EC1	-	SCOPUS	-
Application of humanoid robots to building and home management services	2003	Sawasaki, N.; Nakajima, T.; Shiraishi, A.; Nakamura, S.; Wakabayashi, K.; Sugawara, Y.; , Application of humanoid robots to building and home management services, Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on , vol.3, no., pp. 2992- 2997 vol.3, 14-19 Sept. 2003	EC1	-	IEEE	-
Policy based management for next generation mobile networks	2003	Iacono, S.; Arneodo, F.; Cardoso, K.; Genet, M.G.; Zeglache, D.; , Policy based management for next generation mobile networks, Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE , vol.2, no., pp.1350-1354 vol.2, 20-20 March 2003	EC1	-	IEEE	-
Share it! - the architecture of a rights-managed network of peer-to-peer set-top-boxes	2003	Walker, J.; Morris, O.J.; Marusic, B.; , Share it! - the architecture of a rights-managed network of peer-to-peer set-top-boxes, EUROCON 2003. Computer as a Tool. The IEEE Region 8 , vol.1, no., pp. 251- 255 vol.1, 22-24 Sept. 2003	EC1	-	IEEE	-

Table A.1: Complete list returned by systematic mapping review.

Snow cover mapping using SPOT VEGETATION with high resolution data: application in the Moroccan Atlas Mountains	2003	Hanich, L.; de Solan, B.; Duchemin, B.; Maisongrande, P.; Chaponniere, A.; Boulet, G.; Chehbouni, G.; , Snow cover mapping using SPOT VEGETATION with high resolution data: application in the Moroccan Atlas Mountains, Geoscience and Remote Sensing Symposium, 2003. IGARSS '03. Proceedings. 2003 IEEE International , vol.4, no., pp. 2829- 2830 vol.4, 21-25 July 2003	EC1	-	IEEE	-
InSAR end-to-end simulation environment	2002	Farhat M., Lauzon F., Trudeau A., Fiset R.. InSAR end-to-end simulation environment. International Geoscience and Remote Sensing Symposium (IGARSS), 2002 pp 1480 - 1482.	EC1	-	SCOPUS	-
Eliciting user preferences using image-based experience sampling and reflection	2002	Intille S., Kukla C., Ma X.. Eliciting user preferences using image-based experience sampling and reflection. Conference on Human Factors in Computing Systems - Proceedings, 2002 pp 738 - 739.	EC1	-	SCOPUS	-
Bidirectional Conversion between XML Documents and Relational Data Bases	2002	Jacinto M.H., Librelotto G.R., Ramalho J.C., Henriques P.R.. Bidirectional Conversion between XML Documents and Relational Data Bases. Proceedings of the International Conference on Computer Supported Cooperative Work in Design, 2002 pp 437 - 443.	EC1	-	SCOPUS	-
A database-centric and web-automatic hypertext application design method	2002	Seng J.-L., Wang I.-P.. A database-centric and web-automatic hypertext application design method. Journal of Computer Information Systems, 2002 pp 91 - 109.	EC1	-	SCOPUS	-
ARKTOS: A knowledge engineering software tool for images	2002	Soh L.-K., Tsatsoulis C.. ARKTOS: A knowledge engineering software tool for images. International Journal of Human Computer Studies, 2002 pp 469 - 496.	EC1	-	SCOPUS	-
A testbed for configuration management policy programming	2002	Van Der Hoek A., Carzaniga A., Heimbigner D., Wolf A.L.. A testbed for configuration management policy programming. IEEE Transactions on Software Engineering, 2002 pp 79 - 99.	EC1	-	IEEE, SCOPUS	-
Character string extraction from color documents	2001	Hase H., Shinokawa T., Yoneda M., Y. Suen C.. Character string extraction from color documents. Pattern Recognition, 2001 pp 1349 - 1365.	EC1	-	SCOPUS	-
The wired for peace project: International diplomacy and the virtual private network	2001	Powers P., Oboronko V.. The wired for peace project: International diplomacy and the virtual private network. IEEE Distributed Systems Online, 2001 pp - .	EC1	-	SCOPUS	-
Design of a graphical input to a decision support system for conflict resolution	2001	Song A., Hipel K.W., Kilgour D.M.. Design of a graphical input to a decision support system for conflict resolution. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 2001 pp 1252 - 1257.	EC1	-	IEEE, SCOPUS	-
Access and flexibility in a changing marketplace-case study: the 2002 Salt Lake City Winter Games	2001	Kingman, S.; Richardson, K.; , Access and flexibility in a changing marketplace-case study: the 2002 Salt Lake City Winter Games, Communications Magazine, IEEE , vol.39, no.7, pp.94-99, Jul 2001	EC1	-	IEEE	-
Environmental accounting at Ricoh	2001	Hatano, H.; Uramoto, H.; , Environmental accounting at Ricoh, Environmentally Conscious Design and Inverse Manufacturing, 2001. Proceedings EcoDesign 2001: Second International Symposium on , vol., no., pp.654-657, 2001	EC1	-	IEEE	-

Table A.1: Complete list returned by systematic mapping review.

Hop on board with safety	2001	Voros, K.; , Hop on board with safety, University/Government/Industry Microelectronics Symposium, 2001. Proceedings of the Fourteenth Biennial , vol., no., pp.91, 2001	EC1	-	IEEE	-
Infrastructure under construction: Continuous improvement and learning in projects	2000	Gieskes J.F.B., Ten Broeke A.M.. Infrastructure under construction: Continuous improvement and learning in projects. Integrated Manufacturing Systems, 2000 pp 188 - 198.	EC1	-	SCOPUS	-
Picture archiving and communication systems project management using web-based tools	2000	Patel S., Levin B., Gac Jr. R.J., Harding Jr. D., Chacko A.K., Wider R., Romlein J.. Picture archiving and communication systems project management using web-based tools. Journal of Digital Imaging, 2000 pp 208 - 210.	EC1	-	SCOPUS	-
Towards a reading of the vindolanda stylus tablets: Engineering science and the papyrologist	2000	Terras M.. Towards a reading of the vindolanda stylus tablets: Engineering science and the papyrologist. Human IT, 2000 pp - .	EC1	-	SCOPUS	-
A primer for understanding and applying data mining	2000	Thuraisingham, B.; , A primer for understanding and applying data mining, IT Professional , vol.2, no.1, pp.28-31, Jan/Feb 2000	EC1	-	IEEE	-
Insertion of controller-pilot data link communications into the National Airspace System: is it more efficient?	2000	Massimini, P.A.; Dieudonne, J.E.; Monticone, L.C.; Lamiani, D.F.; Brestle, E.A.; , Insertion of controller-pilot data link communications into the National Airspace System: is it more efficient?, Aerospace and Electronic Systems Magazine, IEEE , vol.15, no.9, pp.25-29, Sep 2000	EC1	-	IEEE	-
Optimal maneuver for multiple aircraft conflict resolution: a braid point of view	2000	Hu, J.; Prandini, M.; Sastry, S.; , Optimal maneuver for multiple aircraft conflict resolution: a braid point of view, Decision and Control, 2000. Proceedings of the 39th IEEE Conference on , vol.4, no., pp.4164-4169 vol.4, 2000	EC1	-	IEEE	-
Linear Hash Functions	1999	Alon N., Dietzfelbinger M., Miltersen P.B., Petrank E., Tardos G.. Linear Hash Functions. Journal of the ACM, 1999 pp 667 - 683.	EC1	-	SCOPUS	-
Character string extraction from a color document	1999	Hase, H.; Shinokawa, T.; Yoneda, M.; Sakai, M.; Maruyama, H.; , Character string extraction from a color document, Document Analysis and Recognition, 1999. ICDAR '99. Proceedings of the Fifth International Conference on , vol., no., pp.75-78, 20-22 Sep 1999	EC1	-	IEEE	-
Distribution utility's trade-off decisions in obtaining sources of electricity	1999	Slavickas, R.A.; Alden, R.T.H.; El-Kady, M.A.; , Distribution utility's trade-off decisions in obtaining sources of electricity, Power Delivery, IEEE Transactions on , vol.14, no.4, pp.1495-1503, Oct 1999	EC1	-	IEEE	-
Insertion of controller-pilot data link communications into the national airspace system: is it more efficient?	1999	Massimini, P.A.; Dieudonne, J.E.; Monticone, L.C.; Lamiano, D.F.; Brestle, E.A.; , Insertion of controller-pilot data link communications into the national airspace system: is it more efficient?, Digital Avionics Systems Conference, 1999. Proceedings. 18th , vol.1/17 pp. vol.1, no., pp.5.A.3-1-5.A.3-6 vol.1, Nov 1999	EC1	-	IEEE	-
Integrating automated field design with existing information systems	1999	Underwood, R.C.; Alberty, S.W.; , Integrating automated field design with existing information systems, Rural Electric Power Conference, 1999 , vol., no., pp.C2/1-C2/6, 1999	EC1	-	IEEE	-

Table A.1: Complete list returned by systematic mapping review.

Passengers queue length measurement	1999	Aubert, D.; , Passengers queue length measurement, Image Analysis and Processing, 1999. Proceedings. International Conference on , vol., no., pp.1132-1135, 1999	EC1	-	IEEE	-
Technology and rural development: assessing technology needs of the Southeastern Anatolia Project in Turkey	1999	Oner, M.A.; Basoglu, N.; Ture, E.; , Technology and rural development: assessing technology needs of the Southeastern Anatolia Project in Turkey, Management of Engineering and Technology, 1999. Technology and Innovation Management. PICMET '99. Portland International Conference on , vol.1, no., pp.457 vol.1, 1999	EC1	-	IEEE	-
The significance of telemedicine in a rural emergency department	1999	Stamford, P.; Bickford, T.; Hsiao, H.; Mattern, W.; , The significance of telemedicine in a rural emergency department, Engineering in Medicine and Biology Magazine, IEEE , vol.18, no.4, pp.45-52, July-Aug. 1999	EC1	-	IEEE	-
Delta Algorithms: An Empirical Analysis	1998	Hunt J.J., Vo K.-P., Tichy W.F.. Delta Algorithms: An Empirical Analysis. ACM Transactions on Software Engineering and Methodology, 1998 pp 192 - 214.	EC1	-	SCOPUS	-
Giving RAD a good name	1998	Stapleton Jennifer. Giving RAD a good name. Computer Bulletin (London), 1998 pp 28 - 29.	EC1	-	SCOPUS	-
Integrated design. The scenario-based four year experience	1998	Dennis, N.D., Jr.; Gross, M.A.; Hall, K.D.; Schemmel, J.J.; Knowles, D.R.; , Integrated design. The scenario-based four year experience, Frontiers in Education Conference, 1998. FIE '98. 28th Annual , vol.2, no., pp.920 vol.2, 4-7 Nov. 1998	EC1	-	IEEE	-
Audio/visual information in construction project control	1997	Abudayyeh O.. Audio/visual information in construction project control. Advances in Engineering Software, 1997 pp 97 - 101.	OK	OK	SCOPUS	Video
A multimedia construction delay management system	1997	Abudayyeh O.Y.. A multimedia construction delay management system. Computer-Aided Civil and Infrastructure Engineering, 1997 pp 183 - 192.	EC1	-	SCOPUS	-
A multimedia construction delay management system	1997	Abudayyeh O.Y.. A multimedia construction delay management system. Microcomputers in Civil Engineering, 1997 pp 183 - 192.	EC1	-	SCOPUS	-
Generation of a 3D triangular surface mesh from digitalized data	1997	Archibald I., Bradley C.H.. Generation of a 3D triangular surface mesh from digitalized data. Proceedings of SPIE - The International Society for Optical Engineering, 1997 pp 134 - 141.	EC1	-	SCOPUS	-
Framework for enhancing the quality and effectiveness of computer animation projects	1997	Czuchry Andrew J., Yasin Mahmoud M.. Framework for enhancing the quality and effectiveness of computer animation projects. Industrial Management and Data Systems, 1997 pp 25 - 30.	EC1	-	SCOPUS	-
Range image integration for direct replication of objects	1997	Godin G., Soucy M.-A.A., Boulanger P.. Range image integration for direct replication of objects. Proceedings of SPIE - The International Society for Optical Engineering, 1997 pp 34 - 44.	EC1	-	SCOPUS	-
Business process change: A study of methodologies, techniques, and tools	1997	Kettinger W.J., Teng J.T.C., Guha S.. Business process change: A study of methodologies, techniques, and tools. MIS Quarterly: Management Information Systems, 1997 pp 55 - 79.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

Using videogrammetry and 3-D image reconstruction to identify crime suspects	1997	Klasen L.M., Fahlander O.. Using videogrammetry and 3-D image reconstruction to identify crime suspects. Proceedings of SPIE - The International Society for Optical Engineering, 1997 pp 162 - 169.	EC1	-	SCOPUS	-
In-house vs. off-the-shelf graphic software tools	1997	Martino Linda, Newell Paul Allen. In-house vs. off-the-shelf graphic software tools. Computer Graphics (ACM), 1997 pp 40 - 42.	EC1	-	SCOPUS	-
The FERET (Face Recognition Technology) program	1997	Rauss P.J., Phillips J., Moon H., Rizvi S.A., Hamilton M.K., DePersia A.T.. The FERET (Face Recognition Technology) program. Proceedings of SPIE - The International Society for Optical Engineering, 1997 pp 2 - 11.	EC1	-	SCOPUS	-
Recovery of superquadric primitives from a range image using GA	1997	Tanahashi H., Murakami N., Yamamoto K.. Recovery of superquadric primitives from a range image using GA. Proceedings of SPIE - The International Society for Optical Engineering, 1997 pp 28 - 33.	EC1	-	SCOPUS	-
Supporting diverse activities with digital documents: a pilot study of The Peter F. Drucker Manuscript and Archives Project	1997	Palmer, J.W.; , Supporting diverse activities with digital documents: a pilot study of The Peter F. Drucker Manuscript and Archives Project, System Sciences, 1997, Proceedings of the Thirtieth Hawaii International Conference on , vol.6, no., pp.118-126 vol.6, 7-10 Jan 1997	EC1	-	IEEE	-
A computer-aided system to improve production rates in construction	1996	Christian J., Hachey D.. A computer-aided system to improve production rates in construction. Advances in Engineering Software, 1996 pp 207 - 213.	EC1	-	SCOPUS	-
European project retain: new approach for IBC in teleradiology and PACS based on a full ATM network	1996	Cordonnier Emmanuel, Eichelberg Marco, Piqueras Joaquim, Treguier Catherine, Heautot J-Francois. European project retain: new approach for IBC in teleradiology and PACS based on a full ATM network. IEEE International Conference on Image Processing, 1996 pp 1 - 4.	EC1	-	SCOPUS	-
Automated analysis of nerve-cell images using active contour models	1996	Fok Y.-L., Chan J.C.K., Chin R.T.. Automated analysis of nerve-cell images using active contour models. IEEE Transactions on Medical Imaging, 1996 pp 353 - 368.	EC1	-	IEEE, SCOPUS	-
Software quality programmes: A snapshot of theory versus reality	1996	Hall T.. Software quality programmes: A snapshot of theory versus reality. Software Quality Journal, 1996 pp 235 - 242.	EC1	-	SCOPUS	-
Factors affecting the quality of software project management:an empirical study based on the Capability Maturity Model	1996	Mcguire E.G.. Factors affecting the quality of software project management:an empirical study based on the Capability Maturity Model. Software Quality Journal, 1996 pp 305 - 317.	EC1	-	SCOPUS	-
Design of an object-oriented multimedia database for personalized multimedia news	1996	Quintana Yuri. Design of an object-oriented multimedia database for personalized multimedia news. Canadian Conference on Electrical and Computer Engineering, 1996 pp 282 - 285.	EC1	-	SCOPUS	-
Fox Movietone News Preservation Project: operational and logistic aspects	1996	Redshaw Rebecca, Wetmore R.Evans. Fox Movietone News Preservation Project: operational and logistic aspects. SMPTE Journal, 1996 pp 544 - 546.	EC1	-	SCOPUS	-

Table A.1: Complete list returned by systematic mapping review.

Hypermedia support for collaboration in requirements analysis	1996	Takahashi Kenji, Potts Colin, Kumar Vinay, Ota Kenji, Smith Jeffrey D.. Hypermedia support for collaboration in requirements analysis. Proceedings of the IEEE International Conference on Requirements Engineering, 1996 pp 31 - 40.	EC1	-	SCOPUS	-
Library of congress digital library effort	1995	Becker Herbert S.. Library of congress digital library effort. Communications of the ACM, 1995 pp 66 - .	EC1	-	SCOPUS	-
DICE video conference system and its application in the EUROMIR missions	1995	Koudelka O., Hughes C.D., Horle J., Riedler W., Skipworth B.. DICE video conference system and its application in the EUROMIR missions. Space Communications, 1995 pp 279 - 287.	EC1	-	SCOPUS	-
UC Berkeley's digital library project	1995	Wilensky Robert. UC Berkeley's digital library project. Communications of the ACM, 1995 pp 60 - .	EC1	-	SCOPUS	-
A line sweep thinning algorithm	1995	Fu Chang; Yung-Ping Cheng; Pavlidis, T.; Tsuey-Yuh Shuai; , A line sweep thinning algorithm, Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on , vol.1, no., pp.227-230 vol.1, 14-16 Aug 1995	EC1	-	IEEE	-
A scalable teleradiology information system	1995	Ho, B.K.T.; Taira, R.; Kangaroo, H.; Steckel, R.J.; , A scalable teleradiology information system, Image Management and Communications, 1995., Proceedings of the Fourth International Conference on , vol., no., pp.118-124, 20-24 Aug 1995	EC1	-	IEEE	-
Demonstration of MMACE prototype system for helix TWT design	1995	McDonald, J.A.; , Demonstration of MMACE prototype system for helix TWT design, Plasma Science, 1995. IEEE Conference Record - Abstracts., 1995 IEEE International Conference on , vol., no., pp.148, 5-8 June 1995	EC1	-	IEEE	-
Towards heterogeneous multimedia information systems: the Garlic approach	1995	Carey, M.J.; Haas, L.M.; Schwarz, P.M.; Arya, M.; Cody, W.F.; Fagin, R.; Flickner, M.; Luniewski, A.W.; Niblack, W.; Petkovic, D.; Thomas, J.; Williams, J.H.; Wimmers, E.L.; , Towards heterogeneous multimedia information systems: the Garlic approach, Research Issues in Data Engineering, 1995: Distributed Object Management, Proceedings. RIDE-DOM '95. Fifth International Workshop on , vol., no., pp.124-131, 6-7 Mar 1995	OK	EC1	IEEE	-
Visualization and database tools for YAC and cosmid contig construction	1995	Thomas, S.W.; Rundensteiner, E.A.; Lee, A.J.; , Visualization and database tools for YAC and cosmid contig construction, System Sciences, 1995. Proceedings of the Twenty-Eighth Hawaii International Conference on , vol.5, no., pp.4-13 vol.5, 3-6 Jan 1995	EC1	-	IEEE	-
Scanned well log image and digital curve management	1994	Zainalabedin K.A., Derr M.E., Fenn C.J.. Scanned well log image and digital curve management. Proceedings - Petroleum Computer Conference, 1994 pp 339 - 342.	EC1	-	SCOPUS	-
Customizable tool for the generation of production-based systems	1993	Bittencourt G., Marengoni M.. Customizable tool for the generation of production-based systems. Applications of Artificial Intelligence in Engineering, 1993 pp 337 - 352.	EC1	-	SCOPUS	-
Open systems on the highways	1993	Powell, P.; , Open systems on the highways, Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on , vol., no., pp.1636-1640 vol.2, 1-3 Nov 1993	EC1	-	IEEE	-



Table A.1: Complete list returned by systematic mapping review.

Imaging tool applications for nuclear power plants	1992	Ketchel, J.M.; , Imaging tool applications for nuclear power plants, Nuclear Science Symposium and Medical Imaging Conference, 1992., Conference Record of the 1992 IEEE , vol., no., pp.769-771 vol.2, 25-31 Oct 1992	EC1	-	IEEE	-
The Minnesota Imaging Project research application for understanding an emerging technology	1992	Wanninger, L.A., Jr.; , The Minnesota Imaging Project research application for understanding an emerging technology, System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on , vol.iv, no., pp.410-419 vol.4, 7-10 Jan 1992	EC1	-	IEEE	-
Cognitive tools for locating and comprehending software objects for reuse	1991	Fischer Gerhard, Henninger Scott, Redmiles David. Cognitive tools for locating and comprehending software objects for reuse. Proceedings - International Conference on Software Engineering, 1991 pp 318 - 328.	EC1	-	SCOPUS	-
Generating fuzzy rules by learning from examples	1991	Wang, L.-X.; Mendel, J.M.; , Generating fuzzy rules by learning from examples, Systems, Man and Cybernetics, IEEE Transactions on , vol.22, no.6, pp.1414-1427, Nov/Dec 1992	EC1	-	IEEE	-
An investigation into software maintenance- Perception and practices	1990	Layzell P.J., Macaulay L.. An investigation into software maintenance-Perception and practices. Conference on Software Maintenance, 1990 pp 130 - 140.	EC1	-	SCOPUS	-
An efficient implementation of decomposable parameter spaces	1990	Taylor, R.W.; , An efficient implementation of decomposable parameter spaces, Pattern Recognition, 1990. Proceedings., 10th International Conference on , vol.i, no., pp.613-619 vol.1, 16-21 Jun 1990	EC1	-	IEEE	-
CASE: analysis and design tools	1990	Oman, P.W.; Bowles, A.J.; Mount, R.; Karam, G.; Kalinsky, D.; Tervonen, M.; Bundonis, V.; Fischer, H.; Fish, M.; Longshore, D.; Akiha, N.; , CASE: analysis and design tools, Software, IEEE , vol.7, no.3, pp.37-43, May 1990	EC1	-	IEEE	-
Excellerator: custom CMOS leaf cell layout generator	1989	Poirier, C.J.; , Excellerator: custom CMOS leaf cell layout generator, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol.8, no.7, pp.744-755, Jul 1989	EC1	-	IEEE	-
An overview of the Iris object-oriented DBMS	1988	Fishman, D.H.; , An overview of the Iris object-oriented DBMS, Comcon Spring '88. Thirty-Third IEEE Computer Society International Conference, Digest of Papers , vol., no., pp.177-180, Feb. 29 1998-March 3 1988	EC1	-	IEEE	-
DATA/IMAGE LOCAL NETWORK: PRIORITY RANDOM SPLITTING WITH CONFLICT DETECTION.	1986	Huang Jian-Cheng, Yeh Chia-Lung. DATA/IMAGE LOCAL NETWORK: PRIORITY RANDOM SPLITTING WITH CONFLICT DETECTION.. Proceedings - IEEE INFOCOM, 1986 pp 510 - 515.	EC1	-	SCOPUS	-

## A.6 Data collected from the selected papers

Publication Information	
<b>Title</b>	Efficient image-aware version control systems using GPU
<b>Author</b>	da Silva Junior J.R., Clua E., Murta L.
<b>Published</b>	2015
<b>Reference</b>	da Silva Junior J.R., Clua E., Murta L.. Efficient image-aware version control systems using GPU. Software - Practice and Experience, 2015 pp - .
Summary	
<p>In this paper, the author introduces an infrastructure to support version control of image artifacts. Due to the amount of data that must be processed, the solution is implemented using a GPU architecture, allowing a massively parallel approach for version control, reaching a speedup over 55 X if compared to the same implementation in CPU. All version control operations (diff, patch, and merge) are implemented by the author. Besides that, the approach does not require any specific tool to be used and act directly over the image file. Additionally, some image transforms can be detected, being the delta composed of just this transform information.</p>	
SQ1: Which techniques are being used?	
<p>The paper uses an approach that works directly at image files, without imposing tools to the user in order to benefit from image version control.</p>	
SQ2: Which tool support is offered?	
<p>The author released a tool that can be used to perform version control over image artifacts.</p>	
SQ3: Does it integrates with existing VCS?	
<p>Yes, a plugin to be integrated to Mercurial do exist.</p>	
SQ4: What intervention degree is necessary by the user?	
<p>It is only necessary for situations where a conflict occur during a merge operation.</p>	
SQ5: How the described proposal has been evaluated?	
<p>It has been evaluated by analysing space consumption and time for processing it both on CPU and GPU. Additionally, space occupied by versioning image over common VCS are evaluated when using and not using the plugin.</p>	
Notes	
<p>The paper presents an interesting approach. It does not require any specific tool to be used and can be easily integrated to an existent repository. Additionally, to boost the processing, the GPU architecture has been employed.</p>	

<b>Title</b>	Inverse image editing: Recovering a semantic editing history from a before-and-after image pair
<b>Author</b>	HU, S.-M. et al.
<b>Published</b>	2013
<b>Reference</b>	HU, S.-M. et al. Inverse image editing: Recovering a semantic editing history from a before-and-after image pair. ACM Trans. Graph., ACM, New York, NY, USA, v. 32, n. 6, p. 194:1–194:11, nov. 2013.

Summary	
In this paper, the author introduces an approach to recover a semantically meaningful editing history from a source image and an edited one. Their technique supports the detection of global and local linear and non-linear color changes, the insertion and removal of objects, and cropping.	
SQ1: Which techniques are being used?	
The proposed architecture is aimed to process and extract semantically information from modifications, such as when some object is moved in a photo. This kind of processing requires heavy processing to recognize objects on photos and track them.	
SQ2: Which tool support is offered?	
This information is not provided by the author.	
SQ3: Does it integrates with existing VCS?	
This information is not provided by the author.	
SQ4: What intervention degree is necessary by the user?	
All processing steps must be guided by the user.	
SQ5: How the described proposal has been evaluated?	
It has been evaluated by analysing both time processing and qualitative change detection.	
Notes	
Although it produced interesting results, their technique requires considerable processing time, and so is not feasible for version control. Besides that, we believe that this approach is tied to be using In Adobe Photoshop.	

<b>Title</b>	A GPU-based architecture for parallel image-aware version control
<b>Author</b>	da Silva, J.R., Pacheco, T., Clua, E., Murta, L.
<b>Published</b>	2012
<b>Reference</b>	da Silva, J.R., Pacheco, T., Clua, E., Murta, L. 2012. A GPU-based architecture for parallel image-aware version control. Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on, 10 pages.
Summary	

In this paper, the author introduces an infrastructure to support version control of image artifacts. Due to the amount of data that must be processed, the solution is implemented using a GPU architecture, allowing a massively parallel approach for version control, reaching a speedup over 55 X if compared to the same implementation in CPU. All version control operations (diff, patch, and merge) are implemented by the author. Besides that, the approach does not require any specific tool to be used and act directly over the image file.

**SQ1: Which techniques are being used?**

The paper uses an approach that works directly at image files, without imposing tools to the user in order to benefit from image version control.

**SQ2: Which tool support is offered?**

The author released a tool that can be used to perform version control over image artifacts.

**SQ3: Does it integrates with existing VCS?**

Yes, a plugin to be integrated to Mercurial do exist.

**SQ4: What intervention degree is necessary by the user?**

It is only necessary for situations where a conflict occur during a merge operation.

**SQ5: How the described proposal has been evaluated?**

It has been evaluated by analysing space consumption and time for processing it both on CPU and GPU.

**Notes**

The paper presents an interesting approach. It does not require any specific tool to be used and can be easily integrated to an existent repository. Additionally, to boost the processing, the GPU architecture has been employed.

<b>Title</b>	Nonlinear revision control for images
<b>Author</b>	Chen, H. T., Wei, L. Y., Chang, C. F.
<b>Published</b>	2011
<b>Reference</b>	Hsiang-Ting Chen, Li-Yi Wei, and Chun-Fa Chang. 2011. Nonlinear revision control for images. ACM Trans. Graph. 30, 4, Article 105 (July 2011), 10 pages.

**Summary**

In this paper the author introduces a GIMP plugin for performing image version control. According to the author, depending upon GIMP editor to perform image version control is considered a negative point. As a consequence, only modifications performed through GIMP over images can be versioned. As the main characteristic, the author represents image modifications through a directional acyclic graph. Linear and nonlinear operations are covered by this plugin. Unfortunately, only one image artifact can be versioned each time.

**SQ1: Which techniques are being used?**

The proposed approach uses the command history offered by GIMP to allow merge and diff over images. The GIMP history can be used to construct an already performed command graph over images, allowing the merge operation by navigation through the graph nodes.

**SQ2: Which tool support is offered?**

A plugin is made available by the author in order to be used in GIMP. This plugin allows a graph generation of modifications over image, which allows the diff, patch, and merge operations to be performed. Besides that, the plugin allows visualizing the generated graph for each image.

**SQ3: Does it integrates with existing VCS?**

Due to the fact that just a plugin is available, this approach is restricted to GIMP editor. So, any form of VCS integration is available.

**SQ4: What intervention degree is necessary by the user?**

The user is responsible for explicitly executing each operation through the graph node selection.

**SQ5: How the described proposal has been evaluated?**

Only the necessary space to store each image version was evaluated.

**Notes**

The paper presents an interesting approach. However, its usage is only possible for GIMP users. The majority of graphical tool uses normally prefer Photoshop over GIMP. Besides that, any processing performance evaluation was made.

<b>Title</b>	Generating photo manipulation tutorials by demonstration
<b>Author</b>	GRABLER, F. et al.
<b>Published</b>	2009
<b>Reference</b>	GRABLER, F. et al. Generating photo manipulation tutorials by demonstration. In: ACM SIGGRAPH 2009 Papers. New York, NY, USA: ACM, 2009. (SIGGRAPH '09), p. 66:1–66:9.
<b>Summary</b>	
In this paper, the author proposes a system to automatically manipulate photos using GIMP and creating visually appealing tutorials. Unfortunately, this approach requires GIMP to record all the transformations applied in its command sequences, thus not allowing its inclusion in any external VCS. This restriction may impose a barrier among users that is not flexible to change its tool or even do not want to.	
<b>SQ1: Which techniques are being used?</b>	
The proposed approach uses the command history offered by GIMP to manipulate transforms made over images. However, it is not aimed to provide image versioning, although merge and diff over images is available.	
<b>SQ2: Which tool support is offered?</b>	

No tool is available.
<b>SQ3: Does it integrates with existing VCS?</b>
No integration is available.
<b>SQ4: What intervention degree is necessary by the user?</b>
The user is responsible for explicitly executing each operation through the graph node selection.
<b>SQ5: How the described proposal has been evaluated?</b>
Due to the fact that it is not aimed to image versioning, no evaluation is performed.
<b>Notes</b>
This papers present some techniques that can be applied for image versioning such as image transforms and recovery. However, this paper is aimed to provides an easy way to build tutorial from images.

Publication Information	
<b>Title</b>	Audio/visual information in construction project control
<b>Author</b>	Abudayyeh, O.
<b>Published</b>	1997
<b>Reference</b>	Abudayyeh O., 1997, "Audio/visual information in construction project control", Advances in Engineering Software, v. 28, n. , pp. 97 - 101.
Summary	
Although not providing a diff, patch, nor merge operation, this paper enforces the necessity for any kind of version control over image and video artifacts. The author simulates this control by using a database and storing a reference to a video or image each time a new aquisition is done. Through this architecture, control information and writing and reading policy history are stored in the database.	
<b>SQ1: Which techniques are being used?</b>	
The proposed approach uses a database to manage video and image artifacts.	
<b>SQ2: Which tool support is offered?</b>	
Not specified.	
<b>SQ3: Does it integrates with existing VCS?</b>	

This information is not provided by the author.

**SQ4: What intervention degree is necessary by the user?**

This information is not provided by the author.

**SQ5: How the described proposal has been evaluated?**

No evaluation was done.

**Notes**

According to images available in the paper, the Microsoft Access database is used. However, due to reduced capability of this database, the number of managed artifacts cannot be high.

## APPENDIX B - Graphics Processing Unit

Graphics processing unit (GPU) is a massively multi-threaded processor that can support, and indeed expects, thousands or even millions of concurrent threads. This way, exposing large amounts of fine-grained parallelism is critical for the efficiency of this architecture.

Due to the growing processing capacity of graphics devices, they became an important choice for massively parallel computation. In the beginning, the GPU programming was only possible by using shaders programs, which required knowledge of the graphics pipeline, such as OpenGL or DirectX, in order to fully explore the device. This scenario has changed by the introduction of CUDA (Computing Unified Device Architecture) [87, 28], a programming language developed by nVidia, which allows GPU programming without requiring knowledge of aspects regarding the real time graphics pipeline architecture.

CUDA is a general purpose parallel architecture used to solve a collection of non graphics problems. For programming, CUDA uses the standard C language, although some wrappers for others languages do exist. CUDA is based on heterogeneous programming, where some code fragments are executed by the CPU and other fragments are executed by the GPU, in a concurrent way.

Graphics devices that support CUDA have hundreds or even thousands of cores that can run up to millions of threads in a fast way. Comparing its processing capacity to CPU, one can see that there is a great advantage of using them instead of CPU for some kinds of problems, as is shown in Figure B.1. While an Intel processor can reach some GFlops, a Titan X GPU can reach over to 7 TFlops, for example.

Programmers often use GPU computation and similar interfaces to accelerate computationally intensive data processing operations, being common to reach fifty times or more performance on the GPU in relation to the same problem solved by a CPU [25].

Modern NVIDIA GPUs are fully programmable many-core chips built around an array of parallel processors [76], as can be seen in Figure B.2. The GPU is composed of an array



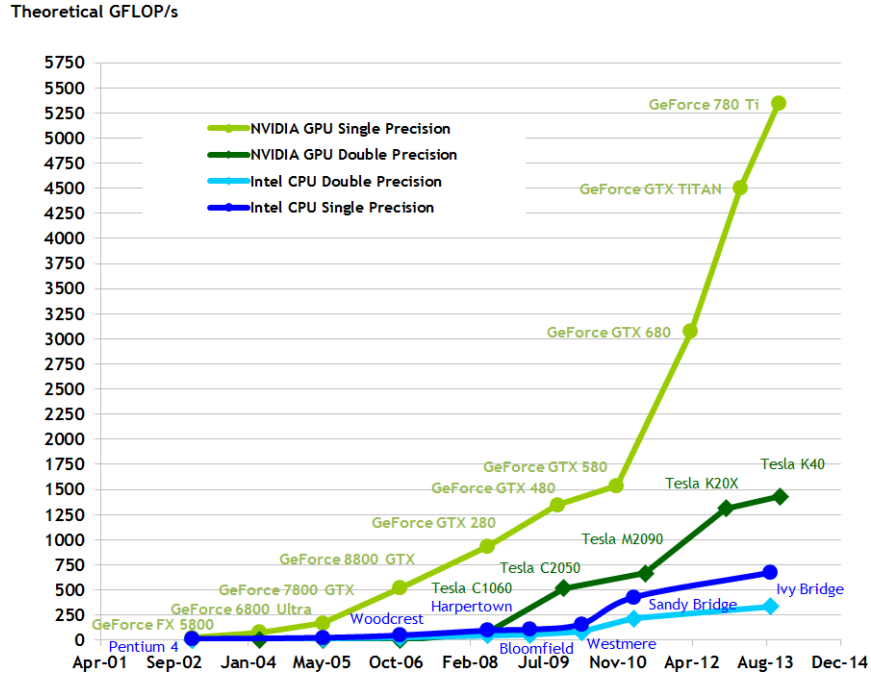


Figure B.1: GPU and CPU processing capacity [28].

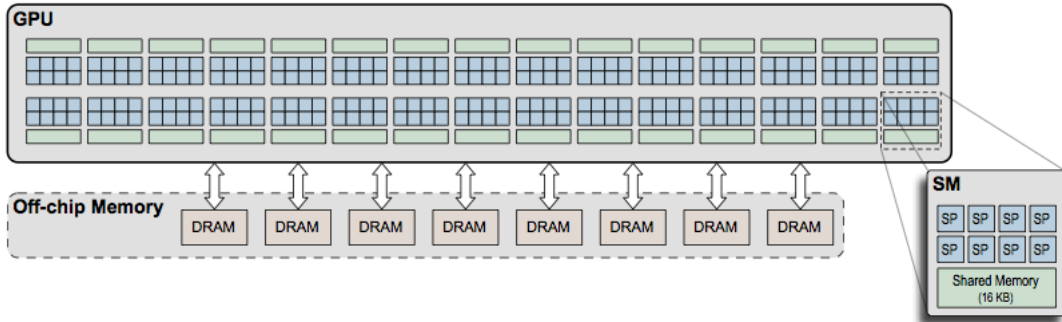


Figure B.2: A common GPU architecture [94].

of Stream Multiprocessor (SM), being each one able to support up to 1024 co-resident concurrent threads. A single SM, as can be seen in Figure B.2, contains many scalar processors (SP), each one with 1024 32-bit registers. Also, each SM is equipped with a shared memory that has very low access latency and high bandwidth, similar to a common L1 cache. The amount of SM and the processors per SM depends on the GPU model.

Additionally, GPUs are composed by the global memory, accessible from both CPU and GPU during processing. This kind of memory is the largest one found on GPU architecture and has the same order of magnitude as a RAM memory in CPUs. Since it is not built in-chip, its latency is large. In fact, using only global memory during processing can slow down the entire system about  $18 \times$ . To avoid this, techniques that uses both

global and shared memory are typically adopted.

Being the GPU an independent device used to alleviate the CPU from graphics processing at first, it is worth to state that GPU and CPU can work asynchronously. This way, the CPU can resume its tasks after starting some processing on GPU. This task distribution is also referred as a heterogeneous architecture. Modern GPUs, such as the *KEPLER* architecture, can also run more than one program at the same time, allowing, for example, the user to play a game while some heavy process is being executed.

Matrices operations are typically very well executed in GPUs. Mapping different data structures and algorithms to matrices consist on an important starting point for porting sequential solutions to GPU parallel ones.

## APPENDIX C - Dominoes Performance Comparision

In order to evaluate Dominoes regarding performance contrasting GPU with CPU, we processed a set of operations using different matrices. The matrices used during this evaluation were extracted using Dominoes, with different time intervals to produce matrices of different sizes. For all the experiment, Apache Derby project has been used. Besides that, all experiments used the [commit|method] ([C|M]) tile. To perform a fair comparison, all linear transformations in CPU were made in OpenBLAS<sup>1</sup>, an open source implementation of BLAS (Basic Linear Algebra Subprograms) API with many handcrafted optimizations for specific processor types, including multi-core parallelism features. An Intel Core i7 4.4GHz PC with 16GB RAM and an nVidia GeForce GTX580 graphics card was used for this evaluation.

In our first experiment, we considered data from a whole year (2013), but varying the number of months for building the [C|M] tile, leading to different matrices sizes. So, in the first interaction, one month was considered while in the last twelve months were used.

Initially, Figure C.1 presents the transposition operation applied over different matrices sizes. As expected, using GPU for processing small matrices tends to be slower when compared to CPU. The main reason for this behavior is due to data transfer from CPU to GPU memory, which is a slow process. The GPU hides this time by leveraging a high number of parallel threads. So, for small matrices, the time of transferring data is higher than the time for processing them. The GPU just becomes faster for matrices size higher than  $261 \times 4135$ .

Besides that, the time for calculating the support is also presented in Figure C.2, where time is presented in a  $\log_{10}$  scale. The support is calculated in Dominos by multiplying a Dominoes tile with its transpose. So, the support for tile [C|M], named [commit|commit] ([C|C]), would be equal to  $[C|C] = [C|M] \times [C|M]^T$ , presenting commits that depend on each other based on dependency over methods.

---

<sup>1</sup>OpenBLAS: <http://www.openblas.net>

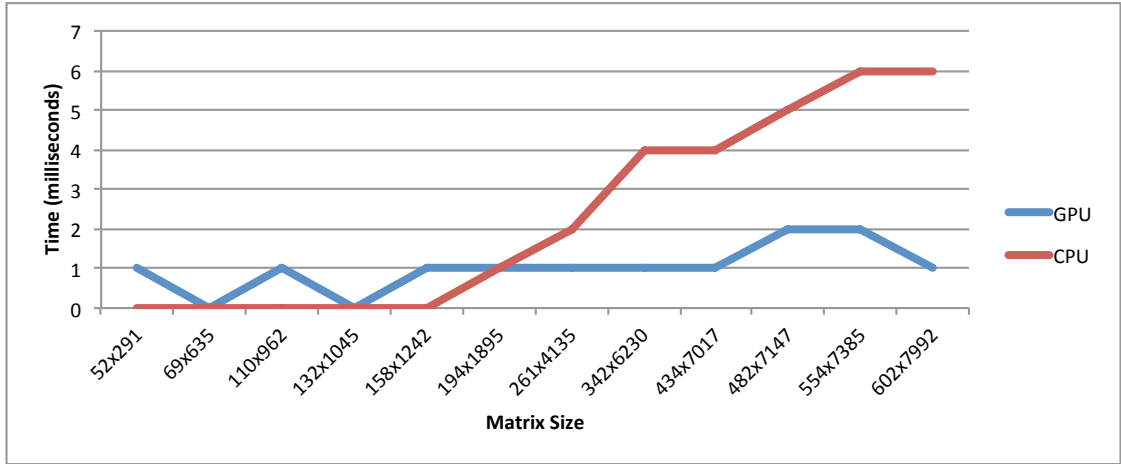


Figure C.1: Transposition operation using different matrix sizes in both CPU and GPU.

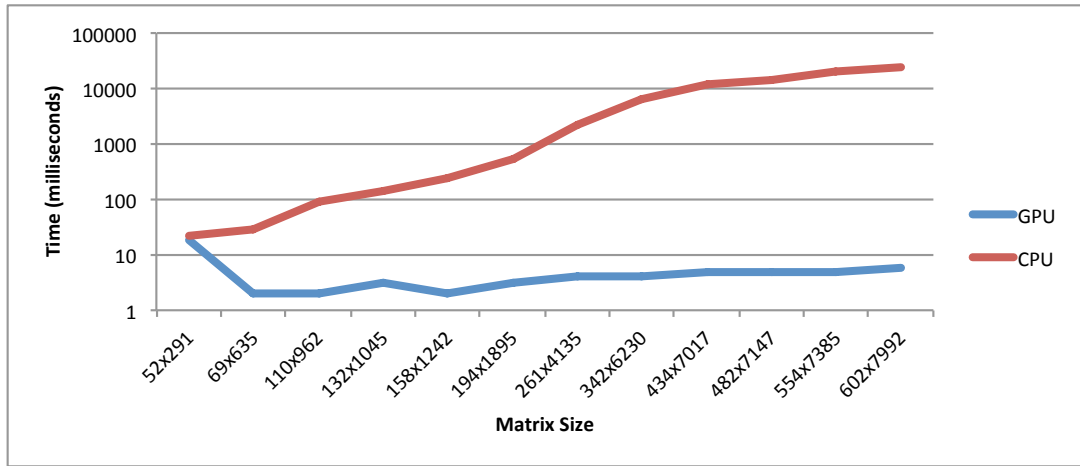


Figure C.2: Support operation using different matrix sizes in both CPU and GPU. Vertical axis are presented in  $\log_{10}$  scale.

From Figure C.2, it is possible to observe that the time for processing support in GPU and CPU is almost the same until the matrix of  $261 \times 4135$ . Since then, processing support operation over CPU increases very fast, while in GPU it keeps almost constant. As said before, the GPU normally becomes efficient when processing large amount of independent data.

Finally, the time for calculating the confidence in CPU and GPU over  $[C|M]$  tile is presented in Figure C.3. In this figure, we can see that GPU requires more time for some matrix sizes. Calculating the confidence requires transferring back and forth data from CPU to GPU and vice versa at different moments. Due to this fact, the main bottleneck for processing confidence is data transfer, which is not significant for larger matrices.

Due to the high amount of time required for processing larger matrices on CPU, we had just used one year for comparing GPU with CPU processing. However, we also

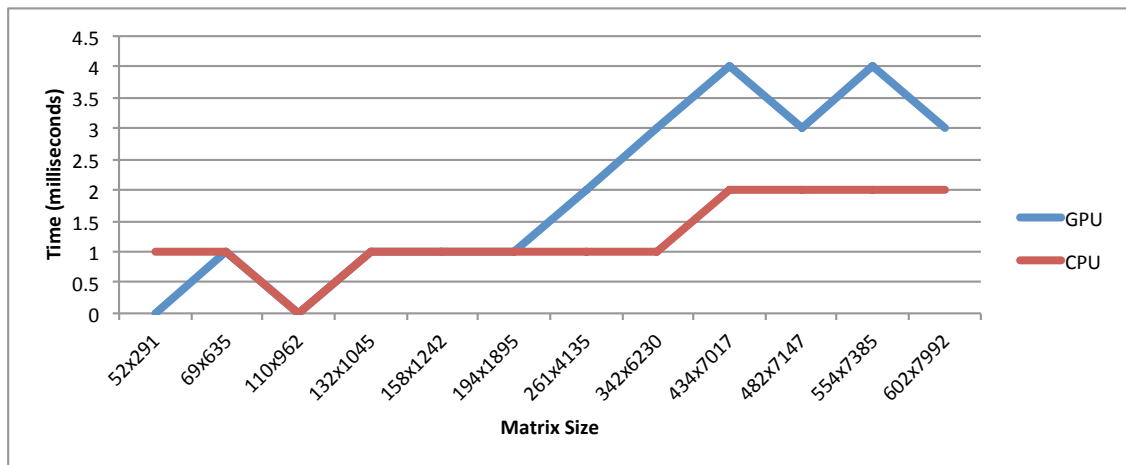


Figure C.3: Confidence operation using different matrix sizes in both CPU and GPU.

processed the whole Derby project on GPU (a period that comprehend data from August 2004 to January 2014), but varying year, as presented in Figure C.4.

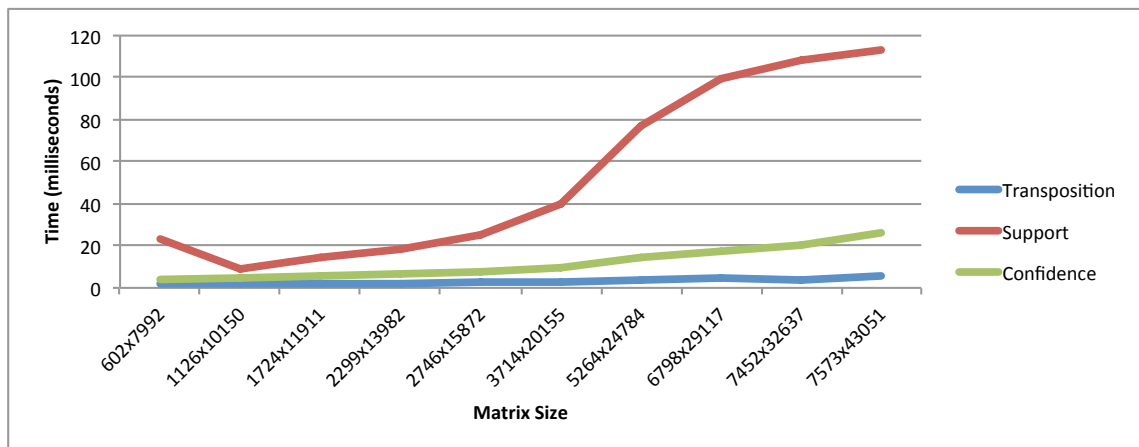


Figure C.4: Processing transpose, support, and confidence operations on GPU using different matrices sizes.

According to Figure C.4, the support operation is the one that requires more time to be processed, reaching about 120 milliseconds for a matrix of  $7,573 \times 43,051$ . On the other hand, both transposition and confidence operation present a slightly increase when the matrix size gets increased.