

JVLC

**Journal of
Visual Language and
Computing**

Volume 2019, Number 1

Copyright © 2019 by KSI Research Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of the publisher.

DOI: 10.18293/JVLC2019-N1

ISSN: 2573-7147 (online)

Journal preparation, editing and printing are sponsored by KSI Research Inc.

Journal of Visual Language and Computing

Editor-in-Chief

Shi-Kuo Chang, University of Pittsburgh, USA

Co-Editors-in-Chief

Gennaro Costagliola, University of Salerno, Italy

Paolo Nesi, University of Florence, Italy

Gem Stapleton, University of Brighton, UK

Franklyn Turbak, Wellesley College, USA

An Open Access Journal published by

KSI Research Inc.

156 Park Square Lane, Pittsburgh, PA 15238 USA

JVLC Editorial Board

Tim Arndt, Cleveland State University, USA
Paolo Bottoni, University of Rome, Italy
Francesco Colace, University of Salerno, Italy
Maria Francesca Costabile, University of Bari, Italy
Philip T. Cox, Dalhousie University, Canada
Martin Erwig, Oregon State University, USA
Andrew Fish, University of Brighton, United Kingdom
Vittorio Fuccella, University of Salerno, Italy
Angela Guercio, Kent State University, USA
Erland Jungert, Swedish Defence Research Establishment, Sweden
Kamen Kanev, Shizuoka University, Japan
Robert Laurini, University of Lyon, France
Jennifer Leopold, Missouri University of Science & Technology, USA
Mark Minas, University of Munich, Germany
Brad A. Myers, Carnegie Mellon University, USA
Joseph J. Pfeiffer, Jr., New Mexico State University, USA
Yong Qin, Beijing JiaoTung University, China
Genny Tortora, University of Salerno, Italy
Kang Zhang, University of Texas at Dallas, USA

Journal Production Associate Editors

Jorge-Luis Pérez-Medina, Universidad de Las Américas, Ecuador
Yang Zou, Hohai University, China

Journal of Visual Language and Computing

Volume 2019, Number 1

September 2019

Table of Contents

A New Beginning for JVLC.	iv
Papers published by the Journal of Visual Languages and Sentient Systems VLSS 2015-2018	vi
Regular Papers	
A Mathematical Language for the Modeling of Geospatial Static Rules. <i>Robert Laurini</i>	1
Context Computation for Implicit Context-Sensitive Graph Grammars: Algorithms and Complexities <i>Yang Zou, Xiaoqin Zeng and Yufeng Liu</i>	15
CROSSIDE: A Design Space for Characterizing Cross-Surface Collaboration by Sketching. <i>Jorge-Luis Pérez-Medina, Jean Vanderdonckt and Santiago Villarreal-Narvaez</i>	29
Collaborative E-learning Environments with Cognitive Computing and Big Data. <i>Mauro Coccoli, Paolo Maresca and Andrea Molinari</i>	43
Comprehension of Software Architecture Evolution Supported by Visual Solutions: A Systematic Mapping and a Proposed Taxonomy. <i>Joao Werther Filho, Glauco Carneiro and Rita Maciel</i>	53

A NEW BEGINNING FOR JVLC

We proclaim a new beginning for the *Journal of Visual Language and Computing*, which is intended to be a forum for researchers, practitioners and developers to exchange ideas and research results, for the advancement of visual languages, visual computing and sentient multimedia systems. Sentient systems are distributed systems capable of actively interacting with the environment by gathering, processing, interpreting, storing and retrieving multimedia information originated from sensors, robots, actuators, websites and other information sources. In order for sentient systems to function efficiently and effectively, visual languages and visual computing play an important role.

The original *Journal of Visual Languages and Computing* was founded in 1980 by me. Together with Dr. Stefano Levialdi, we served as the Founding Co-Editors of JVLC and established JVLC as a high quality, first rate research journal. In 2015 I founded another journal, the *Journal of Visual Languages and Sentient Systems* (VLSS), to encompass the new research area of sentient systems. In 2019 the publisher of JVLC decided not to continue the publication of JVLC. To continue the tradition of JVLC and incorporate the new research area of sentient systems, but to avoid any possible conflicts, I obtained the permission from this publisher to merge VLSS with JVLC and publish the journal under a slightly different name, i.e., *Journal of Visual Language and Computing*. KSI Research Inc., which is a non-profit tax exempt company founded by me to conduct scientific research, organize conferences and publish academic journals and books, became the publisher of the new JVLC.

Although the name change from Visual Languages and Computing to Visual Language and Computing is dictated by necessity, it also connotes a new beginning. The plural form of visual languages suggests unity through diversity. The singular form of visual language suggest diversity through unity, which is consistent with the spirit of merging JVLC and VLSS. In our world, we have witnessed the rapid growth of Internet and sensor-based systems. Thus it is now possible to model the entire world, or at least a significant portion of the world. At the same time the deterioration of the global environment and the depletion of resources call for a deeper understanding of our world by modeling and simulation. Visual languages and visualization will play a very important role in human's efforts to model the world and run simulation experiments, in order to deal with the problems mentioned above. Thus JVLC has both a new beginning and also an urgent mission.

JVLC publishes research papers, state-of-the-art surveys, review articles, in the areas of visual languages, sentient multimedia systems, distributed multimedia systems, sensor networks, multimedia interfaces, visual communication, multi-media communications, cognitive aspects of sensor-based systems, and parallel/distributed/neural computing & representations for multimedia information processing. Papers are also welcome to report on actual use, experience, transferred technologies in sentient multimedia systems and applications. Timely research notes not to exceed ten pages and viewpoint articles, book reviews and tool reviews not to exceed three pages can also be submitted to JVLC.

Manuscripts shall be submitted electronically to jvlc@ksiresearch.org. Original papers only will be considered. Manuscripts should follow the double-column format and be submitted in the form of a pdf file. Page 1 should contain the article title, author(s), and affiliation(s); the name and complete mailing address of the person to whom correspondence should be sent, and a short abstract (100-150 words). Any footnotes to the title (indicated by *, +, etc.) should be placed at the bottom of page 1.

Manuscripts are accepted for review with the understanding that the same work has not been and will not be nor is presently submitted elsewhere, and that its submission for publication has been approved by all of the authors and by the institution where the work was carried out; further, that any person cited as a source of personal communications has approved such citation. Written authorization may be required at the Editor's discretion. Articles and any other material published in JVLC represent the opinions of the author(s) and should not be construed to reflect the opinions of the Editor(s) and the Publisher. For further information contact: jvlc@ksiresearch.org

Shi-Kuo Chang, Editor-in-Chief, JVLC

Papers published by the Journal of Visual Languages and Sentient Systems

VLSS 2015-2018

Volume 1, 2015

Regular Papers

Nurcan Gecer Ulu and Levent Kara, “Generative Interface Structure Design for Supporting Existing Objects”

Gennaro Costagliola, Mattia De Rosa and Vittorio Fuccella, “Fast prototyping of visual languages using local context-based specifications”

Castellano Giovanna, Fanelli Anna Maria and Torsello Maria Alessandra, “Incremental indexing of objects in pictorial databases”

Gennaro Costagliola, Mattia De Rosa and Vittorio Fuccella. RankFrag: A Machine Learning-Based Technique for Finding Corners in Hand-Drawn Digital Curves”

Vincenzo Del Fatto, Vincenzo Deufemia, Luca Paolino and Sara Tumati, “WiSPY: A Tool for Visual Specification and Verification of Spatial Integrity Constraints”

Guoqiang Cai, “GO-Bayes Method for System Modeling and Safety Analysis”

Research Notes

Fabio Pittarello, “Testing a Storytelling Tool for Digital Humanities”

Luca Greco, Francesco Colace, Vincenzo Moscato, Flora Amato and Antonio Picariello, “A Quick Survey on Sentiment Analysis Techniques: a lexical based perspective”

Volume 2, 2016

Regular Papers

Flora Amato, Vincenzo Moscato, Antonio Picariello and Giancarlo Sperli, “Recommender Systems and Social Networks: An Application in Cultural Heritage”

Jennifer Leopold, Nathan Eloie and Chaman Sabharwal, “VisCFSM: Visual, Constraint-Based, Frequent Subgraph Mining”

Gennaro Costagliola, Mattia De Rosa, Andrew Fish, Vittorio Fuccella, Rafiq Saleh and Sarah Swartwood, “A Toolkit for Knot Diagram Sketching, Encoding and Re-generation”

Enrica Pesare, Teresa Roselli and Veronica Rossano, “Visualizing Student Engagement in e-learning Environment”

Paolo Maresca and Andrea Molinari, “Is e-learning Ready for Big Data? And How Big Data Would Be Useful to e-learning ?”

Rita Francese, Carmine Gravino, Michele Risi, Giuseppe Scanniello and Genoveffa Tortora, “Supporting Mobile Development Project-Based Learning by Software Project and Product Measures”

Mehdi Ghayoumi, Arvind Bansal and Maha Thafar, “Towards Formal Multimodal Analysis of Emotions for Affective Computing”

Yanfang Yang and Yong Qin, “Parameter Calibration Method of Microscopic Traffic Flow Simulation Models based on Orthogonal Genetic Algorithm”

Volume 3, 2017

Franklyn Turbak, “Guest Editor’s Introduction to the VLSS Special Issue on Blocks Programming”

Regular Papers

Miguel Ceriani and Paolo Bottoni, “SparqlBlocks: Using Blocks to Design Structured Linked Data Queries”

Michael Homer and James Noble, “Lessons in combining block-based and textual programming”

Michael Knolling, Neil Brown and Amjad Altadmri, “Between blocks and text: A design of program manipulation interactions”

Alexander Repenning, “Moving Beyond Syntax: Lessons from 20 Years of Blocks Programing in AgentSheets”

Michelle Ichinco, Kyle Harms and Caitlin Kelleher, “Towards Understanding Successful Novice Example Use in Blocks-Based Programming”

David Weintrop and Uri Wilensky, “How block-based languages support novices: A Framework for categorizing block-based a_ordances”

Research Notes

Stephanie Ludi and Mary Spencer, “Design Considerations to Increase Block-based Language Accessibility for Blind Programmers Via Blockly”

Volume 4, 2018

Regular Papers

Jennifer Leopold, Nathan Eloie, Jeff Gould and Eric Willard, “A Visual Debugging Aid based upon Discriminative Graph Mining”

Xiaoqin Zeng, Yufeng Liu, Zhan Shi, Yingfeng Wang, Yang Zou, Jun Kong and Kang Zhang, “An Edge-based Graph Grammar Formalism and its Support System”

Gem Stapleton, Lopamudra Choudhury and Mihir Chakraborty, “Spider Diagrams with Absence: Inference Rules for Clutter Reduction”

Soraia M. Alarcão, Ruben Pavão and Manuel J. Fonseca, “Dominant Colors as Image Content Descriptors: A Study with Users”

Soraia M. Alarcão and Manuel J. Fonseca, “Enriching Image Datasets with Unrestrained Emotional Data: A Study with Users”

Shi-Kuo Chang, Cuiling Chen, Wei Guo and Nannan Wen, “Event-Based Data Input, Modeling and Analysis for Meditation Tracking using TDR System”

Research Notes

Walter Balzano and Silvia Stranieri, “A Logic User-Based Algorithm to Improve Node Distribution in Wireless Sensor Network”

Marazzini, Nicola Mitolo, Paolo Nesi and Michela Paolucci, “Smart City Control Room Dashboards: Big Data Infrastructure, from data to decision support”

Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jvlc/

A Mathematical Language for the Modeling of Geospatial Static Rules

Robert Laurini^a

^aKnowledge Systems Institute, USA, and University of Lyon, France

ARTICLE INFO

Article History:

Submitted 4.8.2019

Revised 8.8.2019

Second Revision 5.20.2019

Accepted 8.15.2019

Keywords:

Geographic knowledge

Geographic rules

Formal grammar

Smart cities

ABSTRACT

More and more in information technologies, rules are considered as first-class citizens; and many applications in business intelligence are built on rules. But in territorial intelligence and in smart city planning, few works have been done in this direction. The role of this paper is to show the importance of geographic rules and to propose, beyond the modeling in natural language, a mathematical language to model them. This language is primary based on logic and set theory in which some relations and operators coming from topology, computational geometry and operation research are integrated, and will cover only static rules. By mathematical language, one means that it is independent from any software product or application, and its formal grammar is presented by means of syntactic diagrams. Several examples are provided especially in urban planning.

© 2019 KSI Research

1. Introduction

According to Graham [6] and Morgan [14], rules must be considered as first-class citizens in information technologies, meaning first that several computer-based activities must be revisited. By definition, a rule is a sequence antecedents-implication-consequents which can be noted by $(A) \Rightarrow (B)$, in which A is a conjunction of conditions. Instead of antecedents, sometimes the expression 'premise' is used. In logic, B can be either a disjunction of conditions or a set of assertions. They come from the so-called Horn clauses which are also the basis of logic programming, where it is common to write definite clauses in the form of an implication: $(p \wedge q \wedge \dots \wedge t) \Rightarrow u$.

However, in urban planning, rules are essentially coming from laws and by-laws. Moreover, some experts can use other rules in their daily practice, sometimes

called best practices. In addition to that, more and more specialists in spatial data mining analytics can discover what they call associative rules.

In the knowledge society, in smart city management, it is important not only to identify rules, but also to combine them to automate reasoning. Facing this objective, the scope of this paper is to propose a language in order to model rules. Indeed, rules are often made explicit in natural language; two main sources will be used, namely rules written in natural language and associative rules as extracted from data mining. This language will be based on mathematics without taking into account practical implementation. In other words, this is not a computer language (code). Voluntarily, this paper will not deal with 3D issues, neither with temporal issues: only geographic static rules will be considered.

This paper is organized as follows. First, some elements will be given to explain the role of rules in computing, and especially in geoprocessing. Then the formal grammar of this mathematical language will be detailed. Finally, examples especially in urban planning, will clarify the expressive power of this language.

^aCorresponding author

Email address: Roberto.Laurini@gmail.com

Website: <http://www.laurini.net/robert/>

ORCID: 0000-0003-0426-4030

2. About rules in IT

In this section, the importance of rules will be rapidly emphasized in Business Intelligence. In Business Intelligence, generally, their implementation is based on two grammatical structures IF-THEN-Fact and IF-THEN-Action (Ross [16]). The first serves above all to involve new facts, that is new objects, attribute values, new relationships between objects. As to the second, it is to involve new actions. But who will be in charge of such new actions? In some cases, the computer itself may run procedures or send a message to other devices; in others, particularly in regulatory contexts, a decision maker (for example, the CEO of a company) must himself initiate the action. Another interpretation could be that the choice of alternatives of an action, for example when a law, in some well-defined contexts, opens many perspectives. Thus, a rule is a basic element of a strategy to build reasoning. In contrast to algorithms, they are expressed declaratively. Among business rules, Dietz [3] distinguishes between three categories:

- rejectors typically those related to quality control, that allow a rejection (rejection rules),
- producers such as those determining new values (ex VAT calculation); they can be considered as rules of production of information,
- and projectors such as those related to the replenishment of stocks.

To conclude this section, let us mention that lots of business applications are based on rules, and several computer languages for encoding business rules have been proposed, as XML extensions, such as SWRL (Espinasse, [5]) or RuleML (Boley [1], Boley et al. [2]). The simplest of those extensions is as follows:

```
<Implies>
  <if>
<..>
  </if>
  <then>
    <..>
  </then>
</Implies>
```

Also, very recently, a visual language has been proposed such as by Pittl et al. [15], based on SWRL.

3. Rules in geoprocessing

In contrast, concerning geographic sciences and their applications, few works have been done in the spirit of knowledge engineering. However, let us mention Malerba et al. [13], Salleb-Aouissi et al. [18] and Varadharajulu et al. [22].

From spatial data mining, for instance, Malerba et al. (2003) have discovered the following rule:

$$\begin{aligned} is_a(X, large_town) \wedge intersects(X, Y) \wedge is_a(Y, \\ road) \\ \rightarrow \\ Intersects(X, Z) \wedge is_a(Z, road) \wedge Z \neq Y (91\%, 85\%). \end{aligned}$$

Which states that 'If a large town X intersects a road Y, then X intersects a road Z distinct from Y with 91% support and 85% confidence'. In other words, one can say that *is_a* allows a kind of definition, *intersects* a relation and \neq an additional condition.

Salleb-Aouissi et al. [18] have studied geology and mineral deposits in South America. They extended the existential and universal quantifiers (\forall , \exists) by incorporating buffer zones (here 5 km), and have proposed rules such as:

$$\begin{aligned} Mines - \exists_{5km}^3 Faults \rightarrow True \\ Mines - \exists_{1km}^1 Volcano \rightarrow (active=yes) \end{aligned}$$

The first rule means that there exist at least 3 faults within 5 km of a target object (mineral deposits), whereas the second states that there exists at least one active volcano within 1 km of a target object. They also propose to write $\forall_{5km}^{80\%}$ for considering 80% of items with 5 km. Remark that the distance is taken as a modification of the quantifiers. As it could be nice for colocation rules, other extensions must be defined for other relations coming from topology or computational geometry.

However, Varadharajulu et al. have proposed the following rule for checking road length against road type by using SWRL (remark that the length is given as a data, not computed from computational geometry); more exactly, a road less than 200 m long must be called a "close":

```
NEWRoad(?R1), Roadsuffix(?R1, ?T1),
hasLength(?R1, ?200), SameAs(?T1, ?Close)
-> isAllowed(?R1, true)
```

All those three examples illustrate the difficulties to incorporate issues coming from topology and computational geometry. Indeed, the governance of smart cities must be based on both artificial intelligence and collaborative human intelligence. New domains such as sensors networks, big data, deep learning, geovisualization, etc. must be mobilized together with knowledge engineering to provide more efficient systems for citizens. Do not forget that one of the scopes of urban big data is to discover novel patterns and rules.

3.1 About geographic rules

In Laurini et al. [10], several examples of geographic rules were given, and some important semantical aspects were extracted. Anyhow, geographic rules are a way to model what Shoorchah [21] has called spatial causality.

3.2 Generalities about the model

Based on the previous descriptions, a general diagram for rules can be designed regrouping all aspects, their origin, components, temporal dimension, mathematic tools, their management, their usage and the various modes of implication (Figure 1).

- For physical rules, the implication is mandatory;
- For legal rules, the implication is also mandatory, but the sanctions may or may not exist; in this case, a second rule with the negative conditions will lead to the sanctions;
- For best practices, the implication is more or less a kind of recommendation; in other words, nobody is obliged to follow this kind of rules; perhaps some additional conditions could be considered; at a first approximation, random variates can help to select best practices if any.
- For rules coming from data mining, it is necessary to provide support and confidence. See Shekar et al. [20] for details.

Figure 1 explains the main characteristics of geospatial rules, namely, origin, mathematical tools, temporal dimension, semantics, modality, usage and management. Remark that an existing rule can be superseded by another novel rule, sometimes for a period or a narrower place.

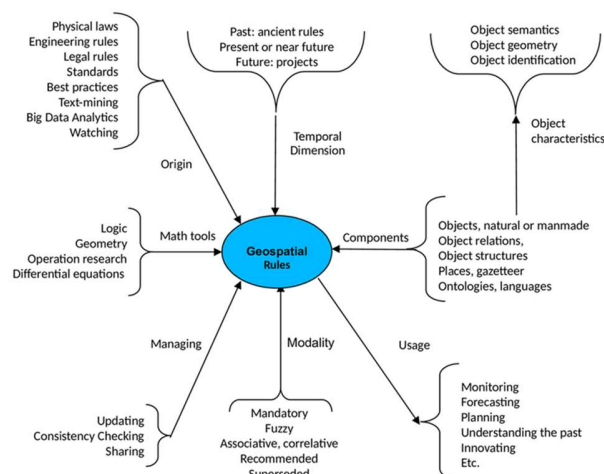


Fig. 1. Main characteristics of geospatial rules.

Now that basic components of geospatial rules are given, let us try to design a mathematical model.

4. A Mathematical Model

This model will be based not only on logics and set theory, but also on computational geometry, topology and fuzzy set theory. Before giving the grammar, some precisions must be done to explain the main components. Anyway, in this model, we assume that:

- a) a global geographic object called *Earth* includes all existing geographic objects (*GO*) and territories (*Terr*),
- b) an ontology will describe their types/classes (whatever is the concept) together with some specific attributes and generic relations between them,
- c) all objects will have valued attributes; in the case of new objects, the attributes are set to null,
- d) there exists a set named *Projects* which comprises all possible environmental and urban projects,
- e) there exists a gazetteer integrating all place names possibly with different variants and in different languages,
- f) it is assumed that all information is correct and consistent,
- g) it is assumed that there are no problems neither regarding geometry accuracy, nor multi-representation,
- h) there are no considerations for storing, implementation, optimization, etc.,
- i) this language is not a rigid language as in computer sciences, but a sketch in which everybody can add symbols, functions, etc.

4.1 Geographic sets and collections

The first elements are the *Earth* and the sets of Geographic Objects (*GO*). As some of them are well characterized (roads, buildings, islands, etc.), others need some clarifications about their definitions. The scope of this paper is not to contribute to those characterizations as works regarding geographic ontologies have revealed several difficulties of categorization. Smith and Varzi [19] have tried to clarify by distinguishing “Fiat” and “Bona Fide” boundaries. Many objects are known by their name (such Sahara), but the boundaries are not well defined, similarly for the Rocky Mountains.

4.2 About places

All places belong to *Earth* and can be described according to various solutions. The simplest ones are by a name, for instance Argentina, the second by a set of coordinates (polygons). Do not forget that a place can form a non-connected polygon (for instance, a country with its islands). More complex solutions can be defined, e.g. the group of countries in which people are driving on

the left, a city district enclosed in a set of streets.

4.3 About geographic objects

All geographic objects (GO) have three types of attributes, for identifying them, for describing them from a geometric point of view and the semantic attributes. The dot notation can be used, for instance "*A.population*". Concerning identification, sometimes ID's can be used, but it is the more common to use a place name or a toponym. As several places can have the same toponym (e.g. Mississippi), the location will resolve ambiguities. Concerning geometry, remember that sometimes an object can have several geometries (multi-representation) sometimes taken at different scales.

Semantic attributes can have different formats such as alphanumeric, Boolean, multimedia, etc. In some cases, fuzzy values can be used such as "near", "far", "low", "high", etc. See Kainz [7] for details and examples.

However, in urban and environmental planning, one needs to consider projects. Indeed, projects have different phases. First, they are designed (design phase) maybe with several sub-steps (preliminary, front-end, etc.). Then there is a legal phase (building permit) in which the project can be approved or rejected. The construction phase itself will followed, sometimes delayed for archaeological or financial reasons.

Finally, the projected object becomes really a geographic feature, finished or not. In other words, it is necessary to consider a *Project* superclass including any geographic object-to be, for instance *Project.Road*. To simplify the problem, the ontology of those objects-to-be will similar to the ontology of existing geographic objects.

4.4 Relations, functions and procedures

Geographic relations are the basis of geographic reasoning. Those relations include Egenhofer et al. [4] 2D topological relations (Figure 2) and other relations.

Some of them are defined for types, and some other for specific geographic objects. Remember that a relation is Boolean (true or false) and can be checked in conditions.

Functions are also useful, some of them coming from computational geometry. To name a few, *Union*, *Intersection*, *Minus*, *Distance*, *Centroid*, *Buffer*, etc., and they will be invoked when necessary. Anyone can add additional functions.

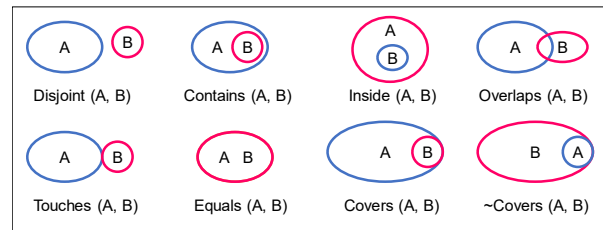


Fig. 2. 2D Topological relations. Egenhofer et al. [4].

Two functions will be extensively used, *Geom* for the geometry and *Topo* for names. By definition *Geom* (*A*) is equivalent to *A.Geom*, and *Topo* (*A*) to *A.Tponym*. Although they are not indispensable, they will be useful in some cases.

Moreover, regarding existing models, the best solution is to encapsulate those mathematical models into procedures for using them when necessary. In addition, a special procedure can be used to call another rule.

4.5 Semantics of Symbols

In this model, the mainstream meaning of mathematical symbol will be accepted. However, due to this special context, some semantics must be fixed for some symbols. Let us explain some problems.

4.5.1 Implications

Three types of implications must be considered. The symbol \Rightarrow will be used for logical assertions coming from physics or legal regulations. For best practices, the expression " \Rightarrow [BP]" will be used, and for associative rules coming from data mining, this symbol will be accompanied with the support and the confidence levels.

4.5.2 About equalities

Remember that the very common sign $=$ has three meanings, (i) for defining something, (ii) for an assignment of value, and (iii) for comparisons. We will use the following symbols:

- a) $=$ as a comparator in Boolean conditions; so that the answer will be either true or false,
- b) \equiv for definitions, especially of new variables,
- c) $:=$ for assigning a new value to a known variable.

4.5.3 The More, the Merrier

Often rules are written according to the style "the more of this, the more of that", or "the less of this, the more of that"; for instance: "the more of traffic, the more of pollution". To solve this problem, three solutions are possible.

- To compare two geographic points – A_1 and A_2 – and to show the evolution in space of some attributes such as if $A_1 > A_2$ then $F(A_1) > F(A_2)$, or $G(A_1) < G(A_2)$,
- A solution can be to write $+x \Rightarrow -y$, the semantics of which are when x increases, then y diminishes; for instance, when saying, the higher, the colder, this sentence can be roughly encoded by $+elevation \Rightarrow -temperature$,
- If a function between both variables is known, it can be used directly.

4.5.4 About homologies

When one needs to compare two geometric shapes, it is commonly accepted that there are some point coordinates which are slightly different. Moreover, for the same object, one person has measured 100 points, whereas another 500. Sometimes, the symbol \approx is used for comparing numerical values. But when we need to compare types and toponyms of geographic objects, other semantics must be ascertained. Indeed, for strings of characters, the Levenshtein distance [12] is used. For instance, between Iraq and Iran the Levenshtein distance is 1, whereas they are very different geographic objects. Regarding types, one can consider that the concepts *street* and *road* are similar. To solve this problem, I have proposed to use the symbol \sqcap (Laurini, [9, 11]) for homologies. So that, considering two geographic objects, we can write:

- $Geom(A) \sqcap Geom(B)$ for comparing these geometric shapes (hence this is equivalent to $Geom(A) \approx Geom(B)$).
- "London" \sqcap "Londres" (London, UK, is called Londres, both in French and Spanish languages; another example is "Washington D.C." \sqcap "District of Columbia").
- $Type(A) \sqcap Type(B)$ when the concept names are different but are corresponding to the same concept, maybe coming from different ontologies.

4.5.5 Other symbols

In logic, the entailment symbol \models is used to denote an affirmation taking the context into account. By writing $\models Contains(A, B)$, one claims that this relationship must be considered as True. Whereas by writing only $Contains(A, B)$, one declares that he is considering this relationship, but he has no hints whether it is True or False. Regarding the other symbols coming from the set theory ($\forall, \exists, \cap, \cup, \supset, \subset, \in, \vee, \wedge, \oplus$, etc.), they have their common meaning.

4.5.6 Consequents

In logic, remember that a conjunctive list of antecedents implies a disjunctive list of consequents so that $A_1 \wedge A_2 \wedge A_3 \wedge \dots A_i \Rightarrow C_1 \vee C_2 \vee C_3 \vee \dots C_j$ in which both A

and C are Boolean expressions. In the disjunctive list, the semantics are not clear: are all C 's true or only a subset? Moreover, in our case, the consequents are not always Boolean (remember IF-THEN-Fact and IF-THEN-Action). To solve this problem, several decisions must be made: (i) discarding the symbol \vee , and separating consequents by the symbol “;”, meaning that all of them are independent and must be enabled; (ii) to use set bracket parentheses $\{\}$ to delimit several consequents if any.

4.5.7 About complex places

Two points of view can now be considered, according to the set theory and according to topology: we can write either $Ghana \in Earth$ or $Contains(Earth, Ghana)$. By extension, the set of left-driving countries can be defined by either $(UK, Ireland, Japan, \text{etc.}) \in LD_Countries$, or $Geom(LD_Countries) \equiv Union(Geom(UK), Geom(Ireland), Geom(Japan), \text{etc.})$. More generally, to define a place, all computational geometry functions can be used. Concerning streets, it could be important to delimit only a sector, for instance by giving civic numbers; for instance, $Geom(S) \equiv StreetSector(Street_name, Civic_number1, Civic_number2)$.

To delimit a city district by a set of surrounding streets, a special function can be invoked *SurroundedByStreet* (*Street1, Street2, Street3, Street4*, etc.).

4.5.8 Remarks

Here are some additional remarks. First, we need to split antecedents in context and conditions. Indeed, in the previous examples, several things were not explicit, i.e. the considered objects and sets were hinted or hidden: those aspects will be described in the context, maybe including definitions and constraints. Moreover, if one needs to consider a negative rule, only the conditions are denied, but not the constraints.

Secondly, let us enlighten the differences between $a \in b$ and $contains(c, d)$. In those expressions, b is a set, and a belongs to this set in the sense of the set theory, whereas c and d are geometric objects and d is inside c from a topological point of view.

5. Formalism

Any geographic rule will be designed as pictured in Figure 3, containing namely Context, Conditions, Implication and Consequents with the following separators : and ■. Context will describe the main variables; Conditions, one or more Boolean criteria to

follow; Implication, the modality; and Consequents, all possible consequences that are to be run.

The grammar is presented by means of the Railroad Diagram Generator (Rademacher, [17]) which is an interesting way to present formally the syntactic structure of a language.

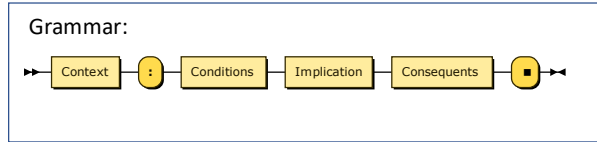


Fig. 3. Formal definition of the grammar.

The Context will be defined as usual from the set theory (Figure 4) by defining variables and their sets, possibly followed by Definitions.

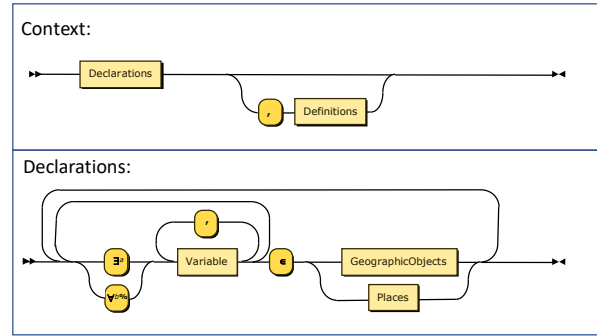


Fig. 4. Context and Declarations.

In Figure 4, as previously explained the extended quantifiers (Salleb-Aoussi et al. [18]) will be used in which a stands either for void (i.e. 1) or for a number greater than 1, whereas b is a percentage. By definition, $\exists^1 = \exists$ and $\forall^{100\%} = \forall$.

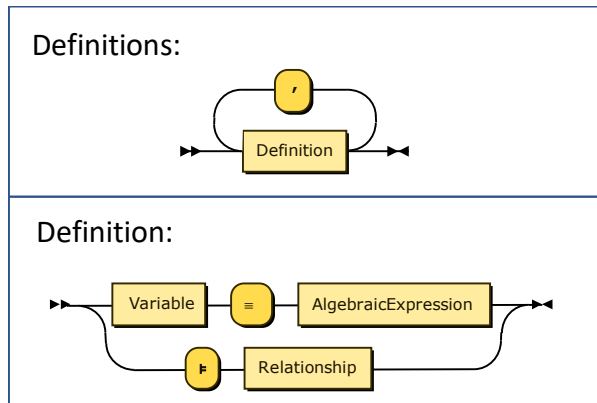


Fig. 5. Definitions and Definition.

Those Definitions (Figure 4) will be used for giving additional statements in the following manner (Figure 5) in which two kinds of statements will be accepted, either the assignment of a value to a variable already defined or the result of an algebraic expression. In this paper, the notion of algebraic expression will not be defined since it is very common in all mathematical languages. In addition, in the Definitions, it is possible to state a relationship between variables considered as a sort of constraint.

Now that the Context is fully characterized, let's consider Conditions. Similarly, Conditions and Condition will be defined in Figure 6.

Regarding implications, as previously told, three modalities exist (Figure 7) in which BP means best practices. As usual, Support and Confidence are percentages for rules coming from data mining.

Now, regarding Consequents, the structure will be similar but recursive (Figure 8).

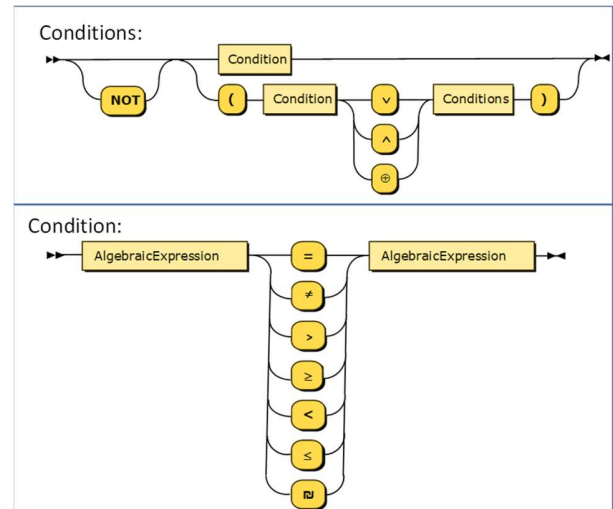


Fig. 6. Conditions and Condition.

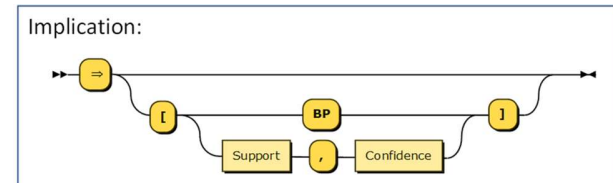


Fig. 7. Implication with three modalities.

In this grammar, the main concepts were presented. Some concepts such as “variable”, “Algebraic Expression”, etc. have their usual meaning in mathematics. Now, let us consider a few examples.

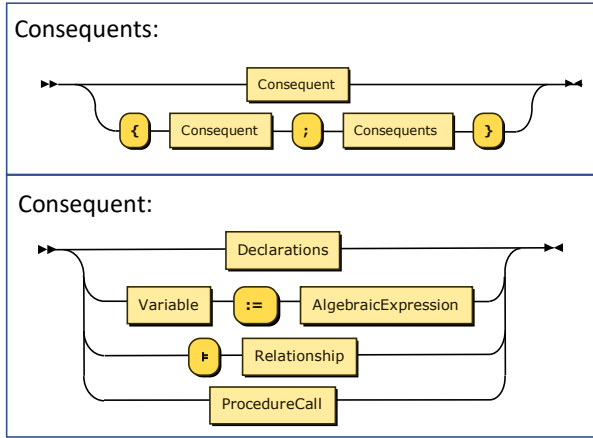


Fig. 8. Consequents and Consequent.

6. Examples

Here are given a few examples of rules written with the above-mentioned language. For that, we suppose the existence of a town named "Smart Town".

6.1 Creating a new relation, a new zone

For defining a new relationship, it can be easily done, for instance for *North* (see Rule 1). An equivalent could be written for *South*. This rule is valid anywhere in the *Earth*.

$\begin{aligned} &\forall p_1, p_2 \in \text{Earth}, \text{GeomType}(p_1) \equiv \text{Point}, \\ &\quad \text{GeomType}(p_2) \equiv \text{Point} \\ &\quad : \\ &\quad \text{Latitude}(p_1) > \text{Latitude}(p_2) \\ &\quad \Rightarrow \\ &\quad \models \text{North}(p_1, p_2) \blacksquare \end{aligned}$	Rule 1
--	--------

For defining *East* and *West*, the rule is a little more complex (Rule 2).

$\begin{aligned} &\forall p_1, p_2 \in \text{Earth}, \\ &\quad \text{GeomType}(p_1) \equiv \text{Point}, \text{GeomType}(p_2) \equiv \text{Point} \\ &\quad : \\ &\quad (\text{Longitude}(p_1) > \text{Longitude}(p_2)) \\ &\quad \wedge (\text{Longitude}(p_1) - \text{Longitude}(p_2) < 180^\circ) \\ &\quad \Rightarrow \\ &\quad \models \text{West}(p_1, p_2) \blacksquare \end{aligned}$	Rule 2
--	--------

Suppose we want to create a new relation when a road is crossing a river. We need to consider a road, a river and their intersection. If there is an intersection between the road and the river, there is a *cross* relationship between them, and it is commutative (Rule 3).

$\begin{aligned} &\forall Ro \in \text{Road}, \forall Ri \in \text{River} \\ &\quad : \\ &\quad \text{Area}(\text{Intersection}(\text{Geom}(Ro), \text{Geom}(Ri))) \neq 0 \\ &\quad \Rightarrow \\ &\quad \{ \models \text{Cross}(Ro, Ri); \models \text{Cross}(Ri, Ro) \} \blacksquare \end{aligned}$	Rule 3
--	--------

For the creation of a flood-risk area in Smart Town, let's supposed it to be defined within a 100 m buffer. Beware, the river can pass either through the town (*Overlap*) or be at the border (*Covers*). Here we must use a constructor, that is a procedure to create a novel geographic object (*CreateGO*). See Rule 4).

$\begin{aligned} &\exists T \in \text{Town}, \forall R \in \text{River}, \text{Topo}(T) \equiv \text{"Smart Town"} \\ &\quad : \\ &\quad \text{Overlap}(R, T) \vee \text{Covers}(T, R) \\ &\quad \Rightarrow \\ &\quad \{ \text{CreateGO}(F); \text{Type}(F) := \text{"Floodplain"}; \\ &\quad \quad \text{Geom}(F) := \text{Intersection}(\text{Geom}(T), \text{Buffer}(R, 100)) \} \blacksquare \end{aligned}$	Rule 4
--	--------

Suppose that a law decides that when a brownfield site is depolluted, it is possible to accept a new building (Rule 5).

$\begin{aligned} &\forall P \in \text{Parcel}, L \in \text{Plants}, B \in \text{Project.BUILDING}, \\ &\quad \models \text{Contains}(P, L), \models \text{Contains}(P, B) \\ &\quad : \\ &\quad L.\text{Erased} \wedge P.\text{Depolluted} \\ &\quad \Rightarrow \\ &\quad \models B.\text{Authorized} \blacksquare \end{aligned}$	Rule 5
--	--------

Suppose now that it was decided not to build in a marsh: this rule can be written as follows (Rule 6):

$\begin{aligned} &\exists C \in \text{County}, M \in \text{Marsh}, \forall B \in \text{Project}, \\ &\quad \models \text{Contains}(C, M) \\ &\quad : \\ &\quad \text{Contains}(M, B) \\ &\quad \Rightarrow \\ &\quad \text{Prohibit}(B) \blacksquare \end{aligned}$	Rule 6
---	--------

6.2 Again, the More, the Merrier

Suppose that "Smart Beach" is a sea resort in which land prices diminish when the distance to the sea increases. This economic rule (Rule 7) can be written as follows when considering two parcels with the same area:

$\begin{aligned} &\forall p_1, p_2 \in \text{Parcel}, \exists S \in \text{Sea}, C \in \text{County}, \\ &\quad \models \text{Touches}(\text{Sea}, C), \\ &\quad \models (\text{Area}(\text{Geom}(p_1)) \sqsupset \text{Area}(\text{Geom}(p_2))) \\ &\quad : \\ &\quad \text{Contains}(C, p_1) \wedge \text{Contains}(C, p_2) \\ &\quad \wedge (\text{distance}(\text{centroid}(\text{geom}(p_1)), S) > \\ &\quad \quad \text{distance}(\text{centroid}(\text{geom}(p_2)), S)) \\ &\quad \Rightarrow \\ &\quad \models p_1.\text{Landprice} < p_2.\text{Landprice} \blacksquare \end{aligned}$	Rule 7
---	--------

With the second solution (Rule 8), we get:

$\begin{aligned} &\forall p \in \text{Parcel}, S \in \text{Sea}, C \in \text{County}, \\ &\quad \models \text{Touches}(\text{Sea}, C) \\ &\quad : \\ &\quad \text{Contains}(C, p) \wedge ++\text{distance}(\text{centroid} \\ &\quad \quad (\text{Geom}(p), S) \\ &\quad \Rightarrow \\ &\quad \models - - p.\text{Landprice} \blacksquare \end{aligned}$	Rule 8
---	--------

Suppose that some economists have evaluated a function F for prices, provided that the area is more than 100 m² and the distance to the sea less than 10 km, the following can be written (Rule 9):

$\begin{aligned} &\forall p \in \text{Parcel}, \exists S \in \text{Sea}, C \in \text{County}, \\ &\quad \models \text{Touches}(\text{Sea}, C), \\ &\quad d \equiv (\text{Centroid}(\text{Geom}(p), S) \\ &\quad : \\ &\quad \text{Contains}(C, p) \\ &\quad \wedge \text{Area}(\text{Geom}(p)) < 100 \\ &\quad \wedge d < 10\,000 \\ &\quad \Rightarrow \\ &\quad p.\text{Landprice} := F(\text{Area}(\text{geom}(p)), d) \blacksquare \end{aligned}$	Rule 9
---	--------

6.3 With fuzzy attributes

In the previous case, another solution can be based on fuzzy values: the previous rule can be transformed into "near the sea, prices are high", and "far from the sea, prices are low" (Rule 10).

$\begin{aligned} &\forall p \in \text{Parcel}, \exists S \in \text{Sea}, C \in \text{County}, \\ &\quad \models \text{Touches}(\text{Sea}, C), \\ &\quad d \equiv (\text{Centroid}(\text{Geom}(p), S) \\ &\quad : \\ &\quad \text{Dist}(S, p) = \text{"near"} \\ &\quad \Rightarrow \\ &\quad p.\text{Landprice} := \text{"high"} \blacksquare \end{aligned}$	Rule 10
---	---------

An additional rule can be written, by respectively replacing "near" and "high" with "far" and "low".

6.4 Best practice rules

Suppose it is a best practice to assign a pollution sensor to each lamppost located in "Churchill Road"; the example is given in Rule 11.

$\begin{aligned} &\exists T, \in \text{Town}, \exists R \in \text{Road}, \forall L \in \text{Lamppost}, \\ &\quad \forall S \in \text{Pollution.Sensor}, \\ &\quad \text{Topo}(T) \equiv \text{"Smart Town"}, \\ &\quad \text{Topo}(R) \equiv \text{"Churchill Road"} \\ &\quad : \\ &\quad \text{Contains}(\text{Geom}(T), \text{Geom}(R)) \wedge \text{Contains} \\ &\quad \quad (\text{Geom}(R), \text{Geom}(L)) \\ &\quad \Rightarrow [\text{BP}] \\ &\quad \text{Assign}(S, L) \blacksquare \end{aligned}$	Rule 11
---	---------

6.5 Associative spatial rules

When studying road incidents in the city of Helsinki, Finland, Karasova et al. [8] have shown by spatial data mining that many incidents occur near bars and restaurants. More exactly, around each incident they have designed a 50 m buffer and see whether there were incidents in those zones (Support 1.7% and confidence 40.0%).

The fact that the support is weak means overall that in their database, there are many other objects, whereas the confidence level means that 40 % of incidents happen in the vicinity of bars or restaurant. This associative rule can be written as follows in which $Terr$ is a territory (Rule 12):

$\begin{aligned} &\exists C \in \text{City}, \forall B \in \text{Bar}, \forall R \in \text{Restaurant}, \\ &\quad \forall I \in \text{Incident}, \\ &\quad \exists \text{RiskyZone} \in \text{Terr}, \text{Topo}(C) \equiv \text{"Helsinki"}, \\ &\quad \models \text{Contains}(\text{Geom}(C), \text{Geom}(B)), \\ &\quad \models \text{Contains}(\text{Geom}(C), \text{Geom}(R)), \\ &\quad \models \text{Contains}(\text{Geom}(C), \text{Geom}(I)), \\ &\quad \text{Geom}(\text{RiskyZone}) \equiv \\ &\quad \text{Union}(\text{Buffer}(\text{Centroid}(\text{Geom}(B), 50), \\ &\quad \quad \text{Buffer}(\text{Centroid}(\text{Geom}(R), 50))) \\ &\quad \Rightarrow [1.7\%, 40.0\%] \\ &\quad \models \text{Contains}(\text{Geom}(\text{RiskyZone}), \text{Geom} \\ &\quad \quad (I)) \blacksquare \end{aligned}$	Rule 12
---	---------

In their study in the city of Antwerp, Belgium, Zhou et al. [23] have a lot of co-location association rules within 600 m buffers.

For instance, they discovered such a rule between kindergartens, playgrounds, but support and confidence were not mentioned (let write α and β for those unknown values in the Rule 13).

$\begin{aligned} &\exists C \in \text{City}, \forall K \in \text{Kindergarten}, \exists P \\ &\quad \in \text{Playground}, \\ &\quad \text{Topo}(C) \equiv \text{"Antwerp"} \\ &\quad : \\ &\quad \text{Contains}(\text{Geom}(C), \text{Geom}(K)), \\ &\quad \text{Contains}(\text{Geom}(C), \text{Geom}(P)) \\ &\quad \Rightarrow [\alpha, \beta] \\ &\models \text{Contains}(\text{Geom}(\text{Centroid}(\text{Geom}(K), 600), \\ &\quad \text{Geom}(\text{Centroid}(P))) \blacksquare \end{aligned}$	Rule 13
---	------------

6.6 Back to previous rules

Let us now rewrite rules given in §3.

The example previously given from Malerba et al. [13] from an Italian census can be modeled by (Rule 14):

$\begin{aligned} &\exists P \in \text{Earth}, \forall X \in \text{LargeTown}, \forall Y, Z \in \\ &\quad \text{Road}, \\ &\quad \text{Topo}(P) \equiv \text{"Italy"} \\ &\quad : \\ &\quad \text{Intersection}(X, Y) \wedge Z \neq Y \\ &\quad \Rightarrow [45\%, 90\%] \\ &\models \text{Intersection}(X, Z) \blacksquare \end{aligned}$	Rule 14
---	------------

The first rule given by Assab et al. [18] ($\text{Mines} - \exists_{5km}^3 \text{Faults} \rightarrow \text{True}$) can be translated into (Rule 15):

$\begin{aligned} &\exists SA \in \text{Earth}, \exists^3 F \in \text{Fault}, \\ &\quad \text{Topo}(SA) \equiv \text{"South America"} \\ &\quad : \\ &\quad \text{Contains}(\text{Geom}(SA), \text{Geom}(F)) \\ &\quad \Rightarrow \\ &\quad \{\exists M \in \text{Mines}; \\ &\quad \models \text{Contains}(\text{Geom}(\text{Buffer}(\text{Centroid}(M), \\ &\quad 5000), \text{Geom}(F))) \blacksquare \end{aligned}$	Rule 15
---	------------

To finish this section, let's consider the ultimate rule designed by Varadharajulu et al. [22] (Rule 16):

$\begin{aligned} &\forall R \in \text{Project.Road} \\ &\quad : \\ &\quad R.Length(R) < 200, \text{Type}(R) = \text{"Close"} \\ &\quad \Rightarrow \\ &\quad \models R.Allowed \blacksquare \end{aligned}$	Rule 16
--	------------

6.7 Located rules

Sometimes rules are located in small places, for example for urban planning zones. Figure 9 illustrates the case in which there are 3 zones in which 5 rules can be applied, those zones being delimited by the list of surrounding streets. Two solutions are possible:

- 1 – for each zone, give the list of applicable rules,
- 2 – for each rule, give the list of zones in which they apply.

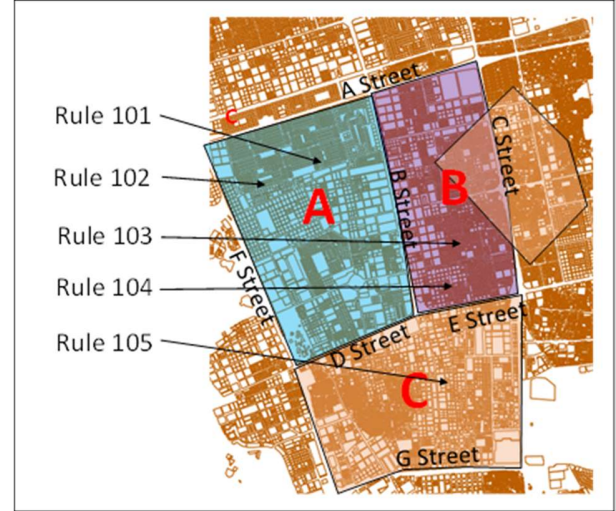


Fig. 9. Rules and zones.

With this rule-oriented language, let's detail take the option that, for each zone, will give the rules to be applied. For zone A, the following can be written (Rule 17):

$\begin{aligned} &\exists C \in \text{City}, \forall B \in \text{Project}, \exists \text{ZoneA} \in \text{Terr}, \\ &\quad \text{Geom}(\text{ZoneA}) \equiv \text{SurroundedByStreet} \\ &\quad (\text{A_Street}, \text{B_Street}, \text{D_Street}, \text{F_Street}) \\ &\quad : \\ &\quad \text{Contains}(\text{Geom}(\text{ZoneA}), \text{Geom}(B)) \\ &\quad \Rightarrow \\ &\quad \{\text{AppliedRule}(101); \text{AppliedRule}(102)\} \blacksquare \end{aligned}$	Rule 17
--	------------

And similar rules can be written for ZoneB and ZoneC.

6.8 Superseded rules

As given in Figure 1, geographic rules can be superseded. But this expression has two meanings. Let us explain them.

The first means that a rule can be replaced by another rule or a set of new rules. For instance, in urban planning when a new plan is adopted. In this case, the previous rule is removed and a new one is enabled. However, geographic objects generated by means of the previous rule still continue to stay; in addition, suppose the previous rule imposes a limitation of 20 m for buildings and the new one only 15 m; it does not imply to demolish those 20m-high existing buildings. Let's call this, temporal superseding.

The second meaning is more complex: suppose that a

new rule is now valid in a small portion of a territory as illustrated in Figure 10. Two solutions are possible; either to state that the new rule (in Z_2) has a priority over the rule in Z_1 ; or to modify the initial rule by replacing $Geom(Z_1)$ by $Minus(Geom(Z_1), Geom(Z_2))$. Let's call this, spatial superseding.



Fig. 10. Example of spatial superseding.

7. Example in Urban Planning

Building permit delivery is a very complex task for local authorities. Indeed, a projected building must be compliant with two types of constraints, national laws and local rules. National laws which are valid everywhere in the country, generally consider structural aspects, such as electricity, insulation, access for handicapped people, etc. – issues which are outside the goal of this paper – whereas local rules consider the location of the building and its conformance to local planning rules, such as height, distance to neighboring parcels, floorspace ratios, etc. Figure 11 depicts the map with the extension of planning zones, and Table 1 gives some limit parameters like maximum height, maximum floorspace ratio and maximum footprint of the building.

TABLE 1. PLANNING ZONES AND THEIR PARAMETERS

Zone ID	Max Height (in m)	Max Floorspace ratio	Max Footprint
Downtown	12	3	80 %
Suburban area	15	4	70 %
Rural area	12	0.5	30 %
Near airport (Bowtie)	8	2	50 %
Airstrip	0	0	0

7.1 Rules for zones

First let's suppose we have rules for dividing the territory and creating zones.

And so, for *ZoneB*, "Suburban area"; *ZoneC*, "Rural

area"; *ZoneD*, "Near airport" and *ZoneE*, "Airstrip", with the appropriate coordinates, respectively numbered Rule 19, Rule 20 and Rule 21. For a projected building supposedly located in the Suburban Area, the rule will be as follows to be accepted.

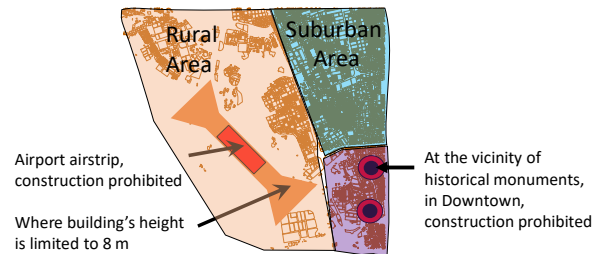


Fig. 11. Planning zones

$\exists C \in City, \exists PZoneA \in Project.Terr,$ $Topo(C) \equiv \text{"Smart Town"}$ $Geom(PZoneA) \equiv Poly(731, 128; 903, 133;$ $905, 341; 839, 346; 814, 349)$ $:$ $Approved(PZone)$ \Rightarrow $AffectName(PZoneA, \text{"Downtown"}) \blacksquare$	Rule 18
--	---------

In this toy-example, we will consider a projected building presented to "Smart Town" local authority, the urban planning department of which has decided to cut the town into three zones, namely downtown, suburban area and rural area.

7.2 Rules regarding parcels and construction

To simplify, we will not consider parcels in which construction is not possible, and parcels for which municipalities may trigger some pre-emptive rights (for instance for schools, parks, etc.).

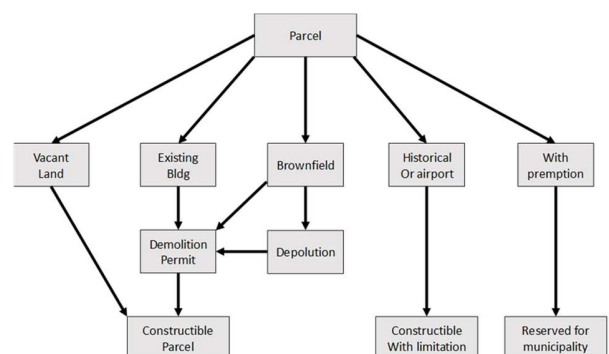


Fig. 12. Different states for a parcel regarding planning.

When there is a pre-existing building, a demolition permit must be granted and when it was a plant

(brownfield), depollution is need (Figure 12). In addition, at the vicinity of a historical monument or of an airport, special limitations apply.

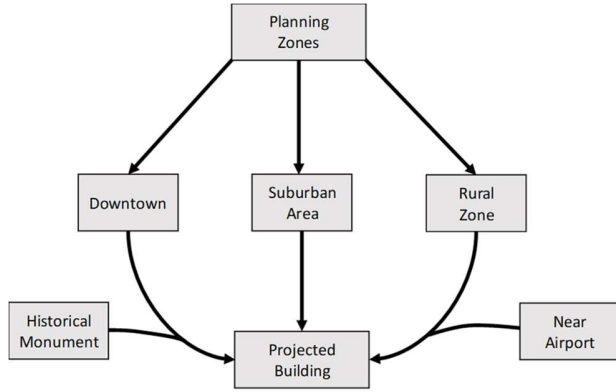


Fig. 13. Any projected building must be compliant with planning rules.

7.3 Projected Buildings and Urban Planning Rules

Regarding planning zones, each has its own regulations. For example, campsites are not allowed in downtown. To simplify, three zones are defined, downtown, suburban area and rural area. However, historical monuments are located in downtown, and it is forbidden to construct new buildings round them (for instance, within 200 meters). In the rural area, there is an airport for which some limitations exist (Figure 11). See Rule 22.

$\begin{aligned} &\exists C \in \text{City}, \exists \text{ZoneB} \in \text{Terr}, \forall B \in \text{Project.Building}, \forall P \in \text{Parcel}, \\ &\text{Topo}(C) \equiv \text{"Smart Town"}, \\ &\text{Topo}(\text{ZoneB}) \equiv \text{"Suburban Area"} \\ &\models \text{Contains}(\text{Geom}(C), \text{Geom}(\text{ZoneB})), \\ &\models \text{Contains}(\text{Geom}(\text{ZoneB}), \text{Geom}(P)), \\ &\models \text{Contains}(\text{Geom}(P), \text{Geom}(B)) \\ &\vdots \\ &B.\text{Height} \leq 15 \\ &\wedge \text{Area}(\text{Union}(\text{Geom}(\text{Floors}))) / \text{Area}(\text{Geom}(P)) \leq 4 \\ &\wedge \text{Area}(B) / \text{Area}(\text{Geom}(P)) \leq 0.70 \\ &\Rightarrow \\ &\models B.\text{ZoneB_Approved} \blacksquare \end{aligned}$	Rule 22
---	---------

But for Downtown and Rural Area, the situation is more complex. For Downtown, we have to take historical monuments into account, for instance within a 200 m buffer around its centroid. See Rule 23.

For the Rural Area, taking the airport into consideration, we need to consider three areas, the area outside the airport, the airstrip in which any building is forbidden, and finally in the “bowtie” with additional limits (Rule 24).

Technically speaking, we have to use an “exclusive or”, noted \oplus between those three possibilities.

$\begin{aligned} &\exists C \in \text{City}, \exists \text{ZoneA}, \text{ConservA} \in \text{Terr}, \forall B \in \text{Project.Building}, \\ &\forall P \in \text{Parcel}, \forall M \in \text{Monuments}, \\ &\text{Topo}(C) \equiv \text{"Smart Town"}, \\ &\text{Topo}(\text{ZoneA}) \equiv \text{"Downtown"}, \\ &\text{Geom}(\text{ConservA}) \equiv \text{Union}(\text{Buffer}(\text{Centroid}(\text{Geom}(M), 200))), \\ &\models \text{Contains}(\text{Geom}(C), \text{Geom}(\text{ZoneB})), \\ &\models \text{Contains}(\text{Minus}(\text{Geom}(\text{ZoneA}), \text{Geom}(\text{ConservA})), \text{Geom}(P)), \\ &\models \text{Contains}(\text{Geom}(P), \text{Geom}(B)) \\ &\vdots \\ &B.\text{Height} \leq 12 \\ &\wedge \text{Area}(\text{Union}(\text{Geom}(\text{Floors}))) / \text{Area}(\text{Geom}(P)) \leq 3 \\ &\wedge \text{Area}(B) / \text{Area}(\text{Geom}(P)) \leq 0.80 \\ &\Rightarrow \\ &\models B.\text{ZoneA_Approved} \blacksquare \end{aligned}$	Rule 23
$\begin{aligned} &\exists C \in \text{City}, \exists \text{ZoneC}, \text{Bowtie}, \text{Airstrip} \in \text{Terr}, \\ &\forall B \in \text{Project.Building}, \\ &\forall P \in \text{Parcel}, \\ &\text{Topo}(C) \equiv \text{"Smart Town"}, \\ &\text{Topo}(\text{ZoneA}) \equiv \text{"Rural Area"}, \\ &\text{Geom}(\text{Bowtie}) = \text{Polyg}(640, 243; 657, 290; 748, 387; 796, 405; 743, 459; 729, 406; 636, 316; 580, 297), \\ &\text{Geom}(\text{Airstrip}) = \text{Polyg}(670, 311; 724, 365; 707, 386; 650, 330), \\ &\models \text{Contains}(\text{Geom}(C), \text{Geom}(\text{ZoneC})), \\ &\models \text{Contains}(\text{Geom}(P), \text{Geom}(B)) \\ &\vdots \\ &(\text{Contains}(\text{Minus}(\text{Geom}(\text{ZoneC}), \text{Geom}(\text{Bowtie})), \text{Geom}(B)), \\ &\wedge B.\text{Height} \leq 12 \wedge \text{Area}(\text{Union}(\text{Geom}(\text{Floors}))) / \text{Area}(\text{Geom}(P)) \leq 0.5 \\ &\wedge \text{Area}(B) / \text{Area}(\text{Geom}(P)) \leq 0.30) \\ &\oplus \text{Disjoint}(\text{Geom}(\text{Airstrip}), \text{Geom}(B)) \\ &\oplus (\text{Contains}(\text{Geom}(\text{Bowtie})), \text{Geom}(B)) \\ &\wedge B.\text{Height} \leq 8 \wedge \text{Area}(\text{Union}(\text{Geom}(\text{B.Floor}))) / \text{Area}(\text{Geom}(P)) \leq 0.5 \\ &\wedge \text{Area}(B) / \text{Area}(\text{Geom}(P)) \leq 0.30) \\ &\Rightarrow \\ &\models B.\text{ZoneC_Approved} \blacksquare \end{aligned}$	Rule 24

Now, to be fully approved by the local administration, whatever its location, a projected building to be approved must follow the subsequent rule; here again, exclusive-ors (\oplus) must be used (Rule 25).

$\begin{aligned} &\exists C \in \text{City}, \forall B \in \text{Project.Building}, \\ &\quad \text{Topo}(C) \equiv \text{"Smart Town"}, \\ &\quad \models \text{Contains}(\text{Geom}(C), \text{Geom}(B)) \\ &\quad \vdots \\ &\quad (\text{Contains}(\text{Geom}(\text{ZoneA}), \text{Geom}(B)) \wedge \\ &\quad \quad \text{B.ZoneA_Approved}) \\ &\oplus (\text{Contains}(\text{Geom}(\text{ZoneB}), \text{Geom}(B)) \wedge \\ &\quad \quad \text{B.ZoneB_Approved}) \\ &\oplus (\text{Contains}(\text{Geom}(\text{ZoneC}), \text{Geom}(B)) \wedge \\ &\quad \quad \text{B.ZoneC_Approved}) \\ &\quad \Rightarrow \\ &\quad \models \text{B.FullyApproved} \blacksquare \end{aligned}$	Rule 25
---	------------

Of course, those rules can be enriched to take depollution for possible brownfields into account as exemplified in Rule 5, flood restriction, etc. Moreover, additional rejection rules can be written.

8. Conclusions

The goal of this paper was to present the first version of a mathematical language for geographic static rule modeling, independent from any computer language and able to integrate all semantics. Thus, this language is based on several mathematical domains such as logic, set theory, computational geometry, topology, etc.

At the moment, only several hundreds of 2D static rules are modeled with an interesting expressive power. For the complete modeling of any geographic rule, research must be carried out in several directions:

- integration of 3D issues, especially for terrain modeling and engineering networks;
- integration of temporal issues; this will lead to dynamic geospatial rules; remember that time semantics are very complex in geography, ranging from fuzzy billions of years in geology until seconds for real time sensors;
- integration of rules deriving from continuous fields, especially for dealing with meteorology, pollution, etc. and other aspects in physical geography;
- integration of additional clauses to extend its expressive power, overall to deal with networks whatsoever, electricity, sewerage, bus lines, etc.;
- looking for more issues in order to enrich semantics, especially for the automatic adaptation to special contexts; for instance, how to adapt a rule such as "when planning a metro, move underground engineering networks" to various street configurations;
- transformation of this mathematical language into a computer language;

- study of metadata relative to geographic rules (origin, modality, etc.);
- design of an inference engine to reason with those rules;
- defining the organization of rules together for their access mechanisms taking temporal and spatial superseding mechanisms.

To conclude this paper, let me thanks the anonymous referees for their work and especially their fruitful comments.

References

- [1] Boley H. (2006) "The RuleML Family of Web Rule Languages". International Workshop on Principles and Practice of Semantic Web Reasoning. Springer, Berlin, Heidelberg, pp. 1-17. Can be downloaded from <http://www.cs.unb.ca/~boley/papers/ruleml-family.pdf>
- [2] Boley, H., Paschke, A., Shafiq, O. (2010) "RuleML 1.0: The Overarching Specification of Web Rules". In Semantic Web Rules: International Symposium, RuleML 2010, Washington, DC, USA, October 21-23, 2010, Proceedings (Vol. 6403, p. 162). Springer Science & Business Media.
- [3] Dietz J.L.G. (2008) "On the Nature of Business Rules". In Advances in Enterprise Engineering, Springer Verlag Lecture Notes in Business Information Processing (10) pp. 1-15.
- [4] Egenhofer M. & Franzosa R.D. (1991) "Point-set topological spatial relations", International Journal of GIS, vol.5, no.2, pp. 161-174.
- [5] Espinasse B. (2017). "Introduction to Semantic Web Rule Language -SWRL". Can be downloaded from <http://www.lsis.org/espinasseb/Supports/ONTOWS/SWRL.pdf>
- [6] Graham I. (2006) "Business Rules Management and Service Oriented Architecture: A Pattern Language". London, John Wiley.
- [7] Kainz W. (2002) "Fuzzy Logic and GIS". Can be downloaded from https://homepage.univie.ac.at/Wolfgang.Kainz/Lehrveranstaltung/en/ESRI_Fuzzy_Logic/File_2_Kainz_Text.pdf
- [8] Karasova V., Krisp J.-M., Virrantas K. (2005) "Application of Spatial Association Rules for Improvement of a Risk Model for Fire and Rescue Services". Proceedings on the 10th Scandinavian Research Conference on Geographical Information Science (ScanGIS), Stockholm, Sweden, pp. 183-193.
- [9] Laurini, R. (2014) "A Conceptual Framework for Geographic Knowledge Engineering", Journal of Visual Languages and Computing (2014), Volume 25, pp. 2-19.
- [10] Laurini R., Servigne S., Favetta F. (2016) "An Introduction to Geographic Rule Semantics". In Proceedings of the 22nd International Conference on Distributed Multimedia Systems (DMS 2016), Salerno, Italy, November 25-26, 2016. Published by Knowledge Systems Institute, ISBN: 1-891706-40-3, pp. 91-97.
- [11] Laurini, R. (2017) "Geographic Knowledge Infrastructure for Territorial Intelligence and Smart Cities". ISTE-Wiley. 250 p.
- [12] Levenshtein, V. I. (1966). "Binary codes capable of correcting deletions, insertions, and reversals". In Soviet Physics, doklady, volume 10, pp. 707-710.
- [13] Malerba D., Esposito F., Lisi, F., Appice A. (2003). "Mining Spatial Association Rules in Census Data. Research in Official Statistics. 5. Can be downloaded from https://www.researchgate.net/publication/2839474_Mining_Spatial_Association_Rules_in_Census_Data/download.

- [14] Morgan T. (2008) "Business Rules and Information Systems: Aligning IT with Business Goals". Addison-Wesley.
- [15] Pittl B., Fill H.-G. (2018) "A Visual Modeling Approach for the Semantic Web Rule Language". *Semantic Web Journal*. September 2018. Can be downloaded from <http://www.semantic-web-journal.net/system/files/swj2023.pdf>
- [16] Ross R. G. (2011) "More on the If-Then Format for Expressing Business Rules: Questions and Answers", *Business Rules Journal*, Vol. 12, No. 4 (Apr. 2011), URL: <http://www.BRCommun2002ity.com/a2011/b588.html>.
- [17] Rademacher G. (2019) "Railroad Diagram Generator". <https://bottlecaps.de/rr/ui>
- [18] Salleb-Aouissi A., Vrain C. & Cassard D. (2015) "Learning Characteristic Rules in Geographic Information Systems". N. Bassiliades, G. Gottlob, F. Sadri, A. Paschke, D. Roman (Eds.). *Rule Technologies: Foundations, Tools, and Applications*, 9th International Symposium, (RuleML 2015), Aug 2015, Berlin, Germany. Springer, 9202, 2015, Lecture Notes in Computer Science.
- [19] Smith B., Varzi A., (2000), "Fiat and Bona Fide Boundaries". *Philosophy and Phenomenological Research* Vol. LX, No. 2, March 2000, pp. 401-420.
- [20] Shekhar S., Huang Y., (2001) "Discovering Spatial Co-location Patterns: A Summary of Results", in *Proc. of the 7th Int'l Symposium on Spatial and Temporal Databases*, 2001, pp. 236–256.
- [21] Shoorcheh M. (2018) "On the spatiality of geographic knowledge". *Asian Geographer*. 36:1, pp. 63-80, DOI: 10.1080/10225706.2018.1463854.
- [22] Varadharajulu P., West G., Mcmeekin D., Moncrieff S. & Arnold L. (2016), "Automating Government Spatial Transactions". In *Proceedings of the 2nd International Conference on Geographical Information Systems Theory, Applications and Management (GISTAM)*, held in April 2016, Rome, Italy, edited by Jorge Gustavo Rocha and Cédric Grueau, published by Scitepress, ISBN: 978-989-758-188-5, pp. 157-167.
- [23] Zhou C., Xiao W.D, Tang D.Q. (2016) "Mining Co-Location Patterns from Spatial Data". *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Volume III-2, 2016 XXIII ISPRS Congress, 12–19 July 2016, Prague, Czech Republic, pp. 85-90.

J

Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jvlc/

Context Computation for Implicit Context-Sensitive Graph Grammars: Algorithms and Complexities

Yang Zou^a, Xiaoqin Zeng, and Yufeng Liu

Institute of Intelligence Science and Technology, School of Computer and Information, Hohai University, China

ARTICLE INFO

Article History:

Submitted 3.1.2019

Revised 6.1.2019

Second Revision 7.22.2019

Accepted 8.26.2019

Keywords:

Visual languages

Context-sensitive graph grammars

Context computation

Algorithm

Complexity

ABSTRACT

Visual Programming Languages have been frequently utilized in computer science. Context-sensitive graph grammars are appropriate formalisms for specifying visual programming languages, since they are intuitive, rigorous, and expressive. Nevertheless, some of the formalisms whose contexts are implicitly or even incompletely represented in productions, called implicit context-sensitive graph grammars, suffer inherent weakness in intuitiveness or limitations in parsing efficiency. Making context explicit to productions tends to be a conceivable way to address this issue. Based on the formalization of context, this paper proposes an approach to the computation of context for implicit context-sensitive graph grammars. The approach is comprised of four partially ordered algorithms. Moreover, the complexities of the algorithms are analyzed and the applicability of the approach is discussed. Thus, the proposed approach paves the way for the practical applications of context in implicit context-sensitive graph grammar formalisms, such as facilitating the comprehension of graph grammars and improving parsing performance of general parsing algorithms.

© 2019 KSI Research

1. Introduction

In many fields of computer science, Visual Programming Languages (VPLs) have been frequently adopted in modeling, representation, and design of complex structures. VPLs usually handle those objects that do not possess inherent visual representation in a visual way [1].

Various approaches have been proposed to formally specify and parsing VPLs, such as constraint multiset grammars [2], symbol-relation grammars [3], picture processing grammar [4], visual grammar [5], attributed shape grammar [6], compiler techniques [7], etc. As a natural extension of formal grammar theory, graph grammars offer the mechanisms for formal specification and parsing of VPLs [8], just like formal grammars do for

string languages. However, the extension from one-dimensional string-based formal grammars to two-dimensional graph grammars brings about a few novel challenges, especially the embedding problem. The embedding problem refers to that how to avoid creating dangling edges when replacing a subgraph in a graph (called host graph) with another graph and connecting the remainder and the replacing graph together to produce a whole graph. Quite a few graph grammar formalisms have been proposed in the literature [8-12]. From the perspective of usability, there is still room for these formalisms to be ameliorated in expressive power or computing efficiency.

Most of the existing graph grammar formalisms fall into the categories of context-free and context-sensitive. The expressive power of a graph grammar lies on the type it belongs to as well as the embedding mechanism it chooses [13-14]. Among all the categories of embedding mechanisms that vary in complexity and power, invariant embedding is the least complex one and most commonly

^aCorresponding author
Email address: yzou@hhu.edu.cn

employed in graph grammar formalisms. Context-sensitive graph grammars tend to be more expressive than context-free ones, when confined to identical less complex embedding mechanisms and invariant embedding in particular. As context-free graph grammars have difficulty in specifying a large portion of graphical VPLs [11-12], recent research in this field focus more on context-sensitive graph grammar formalisms and their applications [15-22].

Generally, a graph grammar consists of a set of productions (rewriting rules), each of which is a pair of graphs, called left graph and right graph, together with an embedding expression. In context-sensitive graph grammars, the contexts pertaining to a production generally refer to the neighboring subgraphs of the rewritten portion of its left graph in potential host graphs [23], which describe the situations under which the production can be applied. A host graph is a graph that is being rewritten by some graph grammar in the process of derivation or parsing. However, the context portion of a production, i.e., the remainder of the left graph minus the rewritten portion, is commonly not a direct copy of the contexts for the sake of conciseness of productions and easiness of embedding.

As is known, the most representative context-sensitive graph grammar formalisms are Layered Graph Grammar (LGG) [11] and Reserved Graph Grammar (RGG) [12]. In order to solve the embedding problem, LGG identically involves in the left and right graphs of a production its immediate context and imposing a dangling edge condition on redex definition, which guarantees that dangling edges never occur in rewritten host graphs. Generally, a redex is a subgraph in a host graph that is isomorphic to the left or right graph of a production. RGG is commonly viewed as an improvement over LGG in respect of succinctness of specification and efficiency of parsing algorithm. Rather than directly involving contexts in productions just as LGG does, RGG formalism invents a particular two-level node structure coupled with a marking technique to indirectly specify the context of a production by identically distributing a set of marked vertices into the left and right graphs. The vertices establish a one-to-one correspondence between the two graphs in terms of their marks. Thus, the embedding problem is solved through this mechanism together with a dedicated embedding rule. Other context-sensitive formalisms include Edge-based Graph Grammar (EGG) [23-24], Context-Attributed Graph grammar (CAGG) [25], Contextual Layered Graph Grammar (CLGG) [26], and Spatial Graph Grammars (SGG) [27]. To tackle the embedding problem, EGG identically augment a set of marked dangling edges to both the left and right graphs of a production, whereas CAGG introduces attributes of nodes to establish a correspondence between the two

graphs of a production. CLGG and SGG are extensions of LGG and RGG, respectively. Based on LGG, CLGG supports three extra mechanisms, which can be employed to define more complex VPLs. SGG extends RGG by augmenting its productions with a spatial specification mechanism, with which it can explicitly describe both structural and spatial relationships for VPLs.

According to how the context portion of a production is dealt with, the preceding formalisms fall into two categories: explicit and implicit [28]. The former formalisms indicate those that directly enclose the complete immediate contexts as its context portion in a production, whereas the latter refers to the ones in which the context portion is expressed as specifically tailored (i.e., incomplete) immediate contexts, attributes adhered to the rewritten portion, or even newly introduced graph notations. Apparently, LGG and RGG are typical examples of the former and the latter, respectively.

One of the inherent deficiencies of implicit formalisms is that they are not intuitive, which arises from the fact that the context portion of a production is not the complete immediate contexts. In RGG, the context portion is a set of marked vertices. Vertices are explained to be connecting points of edges, but their exact meaning is left undefined. Therefore, the selection, arrangement and marking of vertices within a node become a challenge in the design of productions. Moreover, as actual immediate contexts are absent in productions, it is rather difficult for users to exactly comprehend the language of a given graph grammar. Noticeably, similar situations also arise in other implicit formalisms.

Making context explicit can be a conceivable way to address the above issues. Obviously, explicit context is helpful in several application scenarios. Context can be employed to make up the deficiency in intuitiveness so as to facilitate the comprehension and design of implicit graph grammars. Furthermore, context can be utilized to reduce the search space in general parsing algorithms by decreasing the times of backtracking through context matching, thus improving the parsing efficiency. In the literature [28], a formal definition of context is presented and the properties are characterized, which provide a solid theoretical foundation for the computation of context. Nevertheless, the formalization of context and its properties is complex, a direct approach for computation is not available. Therefore, an explicit and detailed method for context computation is apparently a necessity for serving the purpose of context usage in application scenarios.

In this paper, on the basis of RGG formalism, an approach to context computation in implicit graph grammar formalisms is proposed. This is a subsequent research work on context, and the technical contributions

are as follows: It presents a concrete approach for context computation, which is comprised of four partially ordered algorithms with each one being dependent on its predecessors. Moreover, it provides the time complexities of the algorithms. Besides, it discusses the applicability of the approach. This method can be generalized to be applicable to other implicit formalisms. Hence, it paves the way for the application of context for the implicit context-sensitive graph grammar formalisms.

The remainder of the paper is organized as follows: Section 2 reviews the RGG formalism and excerpts the formal definition of context. Section 3 proposes an approach that consists of four algorithms to the computation of context. Section 4 addresses the complexities of the algorithms. Section 5 discusses the applicability of the algorithms. Finally, section 6 concludes the paper and proposes future research.

2. Preliminaries

A graph grammar consists of an initial graph and a collection of productions (graph rewriting rules). Each production has two graphs called left graph and right graph respectively, and can be applied to another graph called host graph. Every node in a production is either a terminal or a non-terminal node. A graph grammar defines a graph language composed of those graphs that can be derived from the initial graph by repeated applications of the productions and whose nodes are all terminal ones. A redex is a subgraph in a host graph that is isomorphic to the left or right graph of a production.

2.1 The RGG Formalism

RGG is a context-sensitive graph grammar formalism [9]. It introduces a node-edge format to represent graphs in which each node is organized as a two-level structure, where the large surrounding rectangle is the first level, called super vertex, and other embedded small rectangles are the second level, called vertices. Either a vertex or a super vertex can be the connecting point of an edge. In addition to the two-level node structure, the RGG also introduces a marking technique that divides vertices into two categories: marked and unmarked ones. Each marked vertex of a production is identified by an integer that is unique in the left or right graph where the vertex lies. A production is properly marked if each marked vertex in the left graph has a counterpart marked by the same integer in the right graph, and vice versa.

In the process of a production application, when a redex is matched in a host graph, each vertex that corresponds to a marked vertex in the left or right graph preserves its associated edges connected to nodes outside of the redex, which avoids the appearance of dangling edges during the

subsequent subgraph replacement process provided that an additional embedding rule is also enforced. The embedding rule states that if a vertex in the right (or left) graph of a production is unmarked and has an isomorphic vertex in the redex of a host graph, then all the edges that are connected to the vertex should be completely inside the redex.

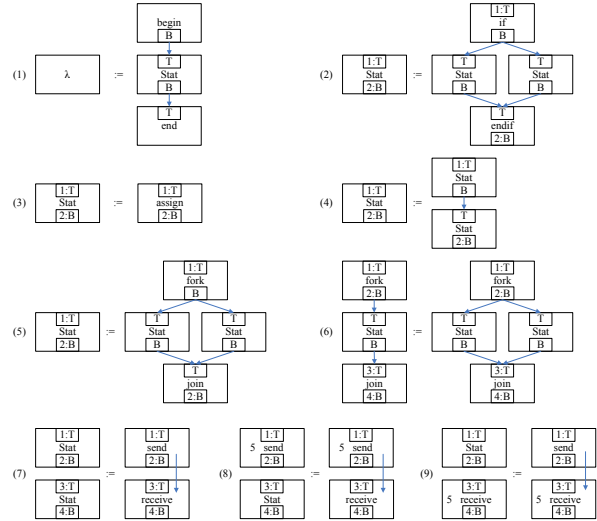


Fig. 1. A graph grammar for process flow diagrams.

As an example, an RGG specifying process flow diagrams, which is slightly adapted from [11], is depicted in Figure 1.

2.2 Partial and Total Precedence

In this subsection, we take the RGG as the representative of implicit context-sensitive graph grammar formalisms to present partial and total precedence relations between graph productions. For the sake of clarity and simplicity, some basic concepts and notations are listed below. Note that graphs are directed ones in the node-edge format and only vertices in productions might be marked.

RGG: A reserved graph grammar is a triple (A, P, Ω) , where A is an initial graph, P a set of graph grammar productions, Ω a finite label set consisting of two disjoint sets Ω^T and Ω^{NT} (called terminal label set and nonterminal label set, respectively). For any production $p := (L, R) \in P$, three conditions are satisfied: R is non-empty, L and R are over Ω , and the size of R are not less than that of L .

$p := (L, R)$: A production with a pair of marked graphs: the left graph L and right graph R . The notations $p.L$ and $p.R$ represent the left and right graph of a production p , respectively. For any graph G , $G.N$ and $G.E$ denote the set of nodes and edges, respectively; $n.V$ and $n.v$ denote the set of vertices and some vertex v of a node n , respectively; and $G.V = \bigcup_{n \in G.N} n.V$ is the union of the sets of vertices of nodes in G ; for any edge e , $s(e)$ and

$t(e)$ represent the source and target vertex of e , respectively, and $l(e)$ is the label on e . $P_L = \{p.L | p \in P\}$ and $P_R = \{p.R | p \in P\}$.

$G_1 \approx G_2$: G_1 is isomorphic to G_2 .

Redex: A subgraph $X \subseteq H$ is a redex of graph G , denoted as $X \in Rd(H, G)$, if $X \approx G$ under an isomorphic mapping f and any vertex in X that is isomorphic to an unmarked vertex in G keeps its edges completely inside X .

$Rd(H, G)$: A set of redexes of marked graph G , which are subgraphs of graph H .

Mcc: A mapping from graphs to the sets of maximal connected components contained in these graphs. A maximal connected component in a graph is a connected component being maximal.

$Mcc(P_L) = \{C | C \in Mcc(p.L) \wedge p \in P\}$, $Mcc(P_R) = \{C | C \in Mcc(p.R) \wedge p \in P\}$.

The following definitions excerpted from [28] are necessary to understand the notion of context and the approach to context computation.

Definition 1. Let $gg := (A, P, \Omega)$ be an RGG, $p_1, p_2 \in P$ be two productions, $C_1 \in Mcc(p_1.L)$ and $C_2 \in Mcc(p_2.R)$. If $\exists X \subseteq C_2$ such that $X \in Rd(C_2, C_1)$, then C_1 is matched with X in C_2 , denoted as $C_1 \approx X \subseteq C_2$; or concisely C_1 is included in C_2 , denoted by $C_1 \sqsubseteq C_2$.

The definition introduces the notion of inclusion between the components of productions, or to be exact, to locate a redex of a component of the left graph of one production in some component of the right graph of another production.

Let U be some set and $S = (B, m)$ a multiset, where B is the underlying set of elements and $m: B \rightarrow \mathbb{N}$ is a mapping from B to the set \mathbb{N} of positive natural numbers. $S \subseteq^{\mathbb{N}} U$ if and only if $B \subseteq U$. In order to unambiguously reference to an element from a multiset, we stipulate that any two elements in a multiset S have distinct identities even if they are the same element from the point of view of the underlying set B , and the identities of elements are not explicitly represented in context for the sake of conciseness.

Definition 2. Let $gg := (A, P, \Omega)$ be an RGG, and P_L and P_R the sets of left and right graphs of productions in P , respectively. A set $S_1 \subseteq Mcc(P_L)$ is included in another multiset $S_2 \subseteq^{\mathbb{N}} Mcc(P_R)$, denoted as $S_1 \sqsubseteq S_2$, if there is a mapping $f: S_1 \rightarrow S_2$ such that:

- $\forall C \in S_1 (\exists X \subseteq f(C) (X \in Rd(f(C), C)))$, and
- $\forall C, C' \in S_1 (C \neq C' \wedge f(C) = f(C') \rightarrow \exists X, X' \subseteq f(C) (X \in Rd(f(C), C) \wedge X' \in Rd(f(C), C') \wedge X \cap X' = \emptyset))$.

In the definition, the first condition states that for each component in S_1 , there is an image in S_2 under the mapping f that contains a redex of it; and the second expresses that if two different components in S_1 have the same image in S_2 , then the two corresponding redexes in it cannot overlap, which strictly adheres to the redex definition in the RGG formalism.

Definition 3. Let $gg := (A, P, \Omega)$ be an RGG, and $p_1, p_2 \in P$ be two productions, p_1 directly partially precedes p_2 , denoted as $p_1 \preceq_d p_2$, if $\exists S \subseteq Mcc(p_2.L)$ such that $S \sqsubseteq Mcc(p_1.R)$. The direct partial precedence relation between them is denoted by the pair $\langle p_1, p_2 \rangle$. The direct partial precedence relation on the set P of productions is defined as $\preceq_P = \{\langle p_1, p_2 \rangle | p_1, p_2 \in P \wedge p_1 \preceq_d p_2\}$. The partial precedence relation between them is denoted by the pair $\langle p_1, p_2 \rangle$.

The direct partial precedence between a pair of productions characterizes the fact that a component of one production's left graph is isomorphic to a subgraph included in a component of another production's right graph.

The partial precedence relation is the closure of the direct partial precedence relation on a set P of productions.

Partial precedence is a kind of relation between a pair of components chosen from two distinct productions, whereas total precedence describes the same relation between two sets of components from the right graphs of a subset of productions and the left graph of a single production, respectively.

Definition 4. Let $gg := (A, P, \Omega)$ be an RGG, $p \in P$, and a multiset $P' \subseteq^{\mathbb{N}} P$. P' directly totally precedes p , denoted as $P' \prec_d p$, if there is a surjective mapping $f: Mcc(p.L) \rightarrow St$ such that:

- $St \subseteq Mcc(P'_R)$;
- $Mcc(p.L) \sqsubseteq Mcc(P'_R)$ with respect to f ;
- $\forall p' \in P' (\exists C \in Mcc(p.L) (f(C) \in Mcc(p'.R)))$.

The corresponding direct total precedence relation is denoted by the pair $\langle P', p \rangle$, and p and P' are called the target production and preceding set, respectively. The direct total precedence relations on the set P of productions is defined as $\prec_P = \{\langle P', p \rangle | P' \subseteq^{\mathbb{N}} P \wedge p \in P \wedge P' \prec_d p\}$.

A direct total precedence relation specifies that a certain graph composed of the right graphs of a subset of productions contains a redex of the left graph of another production. Note that the third constraint on f emphasizes that every production in P' takes part in f with at least one of its components in the right graph. If a subset P' of P forms such a relation with a production p , then it means

that all the right graphs of P' must exactly comprise a redex of the left graph of p with each one containing at least one redex of its components.

A total precedence relation $\langle M, p \rangle$ is composed of a set of direct total precedence relations and a set of linking relations on it. A compound precedence set consists of three parts: a multiset T of productions from set P , a multiset E of direct total precedence relations, and a set R of linking relations on E . A compound precedence set M , together with a production p , forms a total precedence relation, on condition that a direct total precedence relation is established between the first part of M and the production p .

2.3 Definition of Context

The sets of partial or total precedence relations with respect to a graph grammar establish an order of production applications, which can be exploited to discover potential situations in which any of the productions is applicable for derivation. We refer to these situations as contexts. Given two productions p_1 and p_2 , if p_1 directly partially precedes p_2 , then $p_1.R$ contains a context of p_2 or merely a portion of a context, depending on whether $p_2.L$ consists of only one or at least two maximal connected components. As for the former case, $\{p_1\} <_d p_2$ readily holds and a context of p_2 immediately follows; whereas in the latter case, a subset of productions involving p_1 that directly totally precedes p_2 is pursued so as to form a complete context for p_2 . As a third case, a total precedence relation can be sought to build a rather deeper complete context.

Complete contexts of a production can be stratified in terms of the levels of corresponding total precedence relations from which they are generated. Roughly, a complete context that is built in the light of a precedence relation that corresponds to a rooted tree of depth i is called a level- i context, $i \geq 1$, and it degrades to a level-1 context when the relation is a direct one.

A complete context can be employed to extend the respective production to which it pertains. This is done by augmenting the context simultaneously to the both graphs and properly linking the two parts together respectively. A production p equipped with a level- i context is often abbreviated to a context- i p .

Definition 5. Let $gg := (A, P, \Omega)$ be an RGG, $p \in P$, $Mcc(p.L) = \{C_1, \dots, C_n\}$, and $P' \subseteq^N P$. If $P' <_d p$ with respect to some surjective mapping $f: Mcc(p.L) \rightarrow St = \{D_1, \dots, D_m\}$ and a set of redexes $X = \{X_i | X_i \in Rd(f(C_i), C_i), 1 \leq i \leq n\}$, then the pair (U, Z) is a level-1 context of p with respect to P' , f , and X , denoted as $ct_p(P', f, X)$, where $U = \bigcup_{1 \leq j \leq m} D'_j$, $D'_j = D_j \setminus$

$\bigcup_{k_j \in K_j} X_{k_j}$, $K_j = \{l | f(C_l) = D_j \wedge 1 \leq l \leq n\}$, $Z = \bigcup_{1 \leq j \leq m} Z_j$, and $Z_j = \{e \in D_j.E | (s(e) \in X_{k_j}.V \wedge t(e) \in D'_j.V) \vee (s(e) \in D'_j.V \wedge t(e) \in X_{k_j}.V) \wedge k_j \in K_j\}$. The sets U and Z are called the contextual graph and contextual connection, respectively.

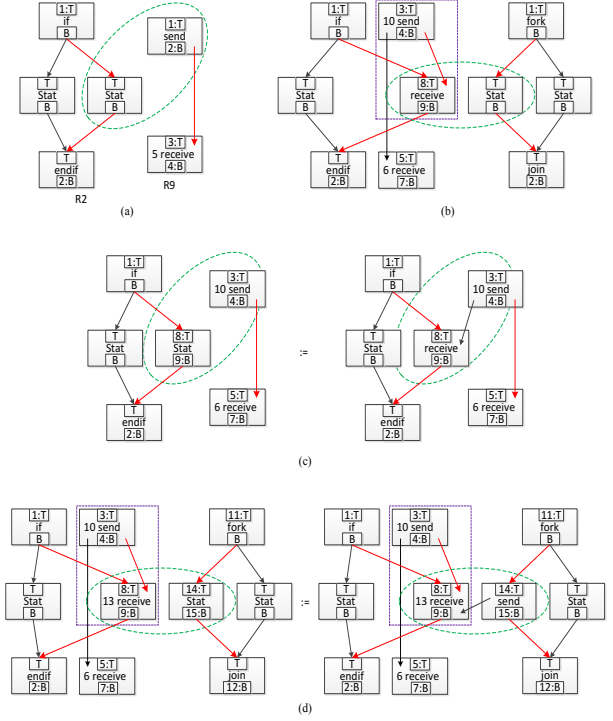


Fig. 2. The contexts of a production and their extended productions. (a) A level-1 context of p_8 . (b) A level-2 context of p_9 . (c) A level-1 context-equipped p_8 . (d) A level-2 context-equipped p_9 .

Each component D'_j of the contextual graph is the rest graph of D_j , a component of the right graph of some production that contains one or more redexes X_{k_j} of the components C_{k_j} , minus X_{k_j} , and each Z_j is the collection of edges in D_j that connect D'_j to all the redexes X_{k_j} .

Similar to Definition 5, the notion of level- i context can be recursively defined.

Example 1. Two contexts of a production and their extended productions.

Two contexts at distinct levels of a production and their respective extended productions are demonstrated in Figure 2. A level-1 context of p_8 originated from the direct total precedence relation $\{p_2, p_9\} <_d p_8$ is shown in Figure 2(a), where the left component of the graph is R_2 (the right graph of production 2), the right one is R_9 , the subgraph enclosed by the green dashed ellipse is the redex of L_8 (the left graph of p_8), and the context consists of two parts U and Z : U is the rest of the whole graph

minus the redex and Z the set of thick red edges that connect U to the redex. The corresponding context-equipped production, context-1 $p8$, numbered as $p11$, is depicted in (c), where the two subgraphs surrounded respectively by green dashed eclipses are the isomorphic image of the underlying production of $p11$.

Figure 2(b) is a level-2 context of $p9$, which is created based on the direct total precedence relation $Ud(\{p5, p11\}) \prec_a p9$, or equivalently, the total precedence relation $Ul(\{p5, p11\}) \prec p9$, where $Ud(P)$ indicates the set of underlying productions of P , and $Ul(P)$ refers to the underlying structure of P . In this graph, the left and right component is $R11$ and $R5$, respectively, the subgraph enclosed by the purple dashed rectangle is the isomorphic image of $Ud(p11)$, and the one surrounded by the green dashed eclipse is the redex of the left graph of $p9$. In a similar way, the corresponding context-equipped production, context-2 $p9$, is illustrated in (d).

3. Context Computation

In this section, an approach is presented for the computation of context in the RGG formalism, based on the theoretical foundation reviewed in the preceding section.

The approach consists of four partially ordered algorithms, of which the first three deal with the computation of the set of direct partial precedence relations, the set of direct total precedence relations, and the set of total precedence relations with respect to a set of productions, respectively, and the last handles the computation of the contexts of a single production as well as the corresponding extended productions.

3.1 Computation of Partial Precedence

Algorithm 1. Computation of partial precedence relations.

Input. A set P of productions.

Output. The direct partial precedence relation on P .

```

{
   $Cm_L = \bigcup_{p \in P} Mcc(p, L)$ ;
   $Cm_R = \bigcup_{p \in P} Mcc(p, R)$ ;
  Create a two-dimensional array  $M$  whose two indices range over
   $Cm_L$  and  $Cm_R$  respectively such that each element is initialized
  to an empty set  $\emptyset$ ;
  for each  $C \in Cm_L$  {
    for each  $C' \in Cm_R$  {
       $M[C, C'] = \text{FindRedex}(C', C)$ ;
    }
  }
  return  $M$ ;
}
```

The first algorithm generates the partial direct precedence relation on a given set of productions. It consists of collecting all the components of the left and

right graphs of the productions to create one set and another respectively, and then finding all the redexes of each component of the former in any one of the latter that is taken as the host graph. The output is a matrix such that each entry is assigned to a set of redexes (maybe empty) with respect to a pair of components from the two distinct sets that uniquely locate the entry in it. The function $\text{FindRedex}(C', C)$ returns all the redex of C found in C' .

For example, consider the RGG in Figure 1, the set of direct partial precedence relations regarding $p8$ is the set that is comprised of the following elements: $\langle p9, p8 \rangle$, $\langle p8, p8 \rangle$, $\langle p7, p8 \rangle$, $\langle p2, p8 \rangle$, $\langle p4, p9 \rangle$, $\langle p5, p9 \rangle$, $\langle p6, p9 \rangle$, $\langle p1, p9 \rangle$, $\langle p8, p9 \rangle$. This set also coincides with the set of partial precedence relations of $p8$.

3.2 Computation of Direct Total Precedence

The second algorithm figures out the direct total precedence relation on a set P of productions, on the basis of the output of the first algorithm.

First, it creates three one-dimensional arrays Llt , Rlt , Tpd and Fun which share a same index that ranges over P , to store the upcoming data.

Then, it arranges the components of the left graph of each production p into an ordered tuple form $Llt[p]$ in a certain order, and for each element in this tuple, it generates a corresponding set that gathers all the redexes involved in the components from Cm_R .

Next, it conducts the Cartesian product $Rlt[p]$ of these sets of redexes in the same order as their counterparts in $Llt[p]$. As a result, every tuple in $Rlt[p]$ is a redex of $Llt[p]$, since they are of the same length and each constituent of the former is a redex of the element of the latter to which it corresponds in terms of the tuple order.

After that, by replacing each redex in a tuple of $Rlt[p]$ with the underlying production whose right graph includes a component that contains this redex, it acquires a direct total precedence relation regarding p ; this process repeats until all the relations regarding p , which comprise the set $Dtp[p]$, are collected. Meanwhile, it establishes a mapping h from $Dtp[p]$ to a partition of $Rlt[p]$ (i.e., a collection of disjoint nonempty subset of it that have it as their union) such that h maps each tuple to the subset of $Rlt[p]$, each of whose elements can be transformed to this tuple by the preceding replacement.

Note that each element in $Dtp[p]$ is a list of a multiset of productions in a predefined order. Nevertheless, a direct total precedence relation refers to a relation between a multiset of productions without any order and a production. To bridge this gap, it performs a partition of $Dtp[p]$ in terms of such an equivalence relation that two lists are equivalent if both of them correspond to a same

multiset, which produces $Dps[p]$ and l .

Algorithm 2. Computation of direct total precedence relations.

Input. A set P of productions and the direct partial precedence relation on P .

Output. The direct total precedence relation on P .

```

{
  Let  $Llt, Rlt, Dtp, Dps, Fun1$  and  $Fun2$  be one-dimensional
  arrays that share the same index which ranges over  $P$ ;
  for each  $p \in P$  {
     $k = |Mcc(p.L)|$ ;
    Let  $Mcc(p.L) = \{C_1, \dots, C_k\}$ ;
     $Llt[p] = \langle C_1, \dots, C_k \rangle$ ;
    for each  $C \in Mcc(p.L)$  {
       $Rdx(C) = \emptyset$ ;
      for each  $C' \in Cm_R$  {
         $Rdx(C) = Rdx(C) \cup M[C, C']$ ;
      }
    }
     $Rlt[p] = Rdx(C_1) \times \dots \times Rdx(C_k)$ ;
     $Dtp[p] = \emptyset$ ;
    for each  $k$ -tuple  $(t_1, \dots, t_k) \in Rlt[p]$  {
      Generate a  $k$ -tuple  $(p_1, \dots, p_k)$  such that
       $t_i \in FindRedex(Mcc(p_i.R), C_i)$ ,  $1 \leq i \leq k$ ;
      if  $((p_1, \dots, p_k) \notin Dtp[p])$  {
         $h((p_1, \dots, p_k)) = \{(t_1, \dots, t_k)\}$ ;
         $Dtp[p] = Dtp[p] \cup \{(p_1, \dots, p_k)\}$ ;
      } else {
         $h((p_1, \dots, p_k)) = h((p_1, \dots, p_k)) \cup \{(t_1, \dots, t_k)\}$ ;
      }
    }
     $Fun1[p] = h$ ;
     $Dps[p] = \emptyset$ ;
    for each list  $(q_1, \dots, q_k) \in Dtp[p]$  {
      Create a multiset  $S = \{q_1, \dots, q_k\}$ ;
      if  $(S \notin Dps[p])$  {
         $l(S) = \{(q_1, \dots, q_k)\}$ ;
         $Dps[p] = Dps[p] \cup \{S\}$ ;
      } else {
         $l(S) = l(S) \cup \{(q_1, \dots, q_k)\}$ ;
      }
    }
     $Fun2[p] = l$ ;
  }
  return  $(Llt, Rlt, Dtp, Dps, Fun1, Fun2)$ ;
}

```

In this way, it finally achieves the arrays Dps and $Fun2$ that can generate the direct total precedence relations regarding each production of P , together with the array $Fun1$ that can produce the set of redexes with respect to any of these relations. For example, for each $S \in Dps[p]$, $\langle S, p \rangle$ is a direct total precedence relation, and $\bigcup_{Q \in l(S)} h(Q)$ is the set of all the possible redexes.

An underlying assumption for the algorithm is that any component in Cm_L or Cm_R , as well as any redex of a component from Cm_L in another from Cm_R , is uniquely identified. This is applicable from the perspective of algorithm implementation, as it can be achieved by assigning to each node of a production a unique number as its identity, and representing each redex as a triple (S, f, id) with S the involved subgraph, i.e., the redex itself, f the underlying mapping, and id the identifier of the right graph that contains the redex.

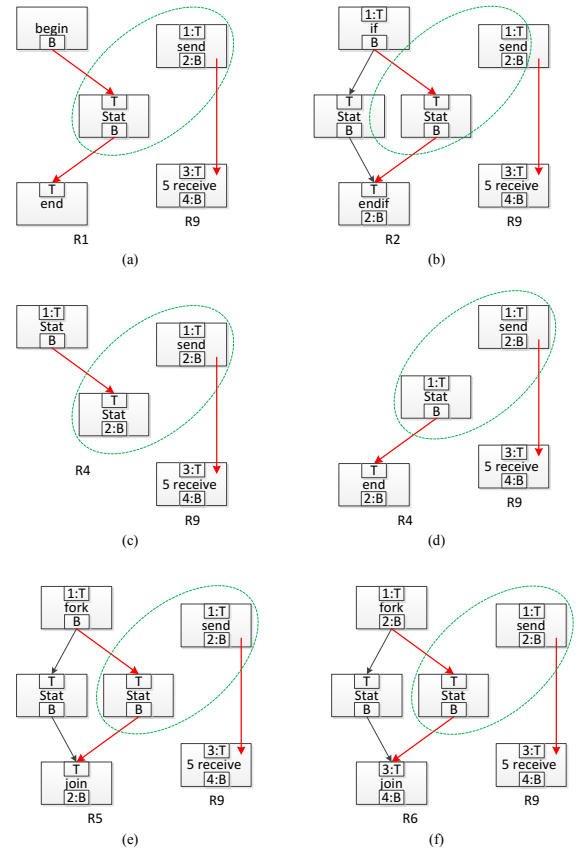


Fig. 3. Some level-1 contexts of $p8$.

For example, consider the RGG illustrated in Figure 1, the direct total precedence relations with $p8$ being the target production are as follows: $\langle \{p1, p9\}, p8 \rangle$, $\langle \{p2, p9\}, p8 \rangle$, $\langle \{p4, p9\}, p8 \rangle$, $\langle \{p5, p9\}, p8 \rangle$, $\langle \{p6, p9\}, p8 \rangle$, $\langle \{p1, p8\}, p8 \rangle$, $\langle \{p2, p8\}, p8 \rangle$, $\langle \{p4, p8\}, p8 \rangle$, $\langle \{p5, p8\}, p8 \rangle$, $\langle \{p6, p8\}, p8 \rangle$, $\langle \{p1, p7\}, p8 \rangle$, $\langle \{p2, p7\}, p8 \rangle$, $\langle \{p4, p7\}, p8 \rangle$, $\langle \{p5, p7\}, p8 \rangle$, $\langle \{p6, p7\}, p8 \rangle$. The preceding set of the relations includes two productions, each of whose right graph contains a component, and these two components constitutes a graph that contains a redex of the target production's left graph. Therefore, the number of total precedence relations regarding $p8$ is $5 \times 3 = 15$. The algorithm figures out all the total precedence relations with any production from the input production set assuming the role of target production.

Figure 3 illustrates some of the level-1 contexts regarding $p8$. These six level-1 contexts correspond to the first five direct total precedence relations with $p8$ being the target production. Note that $\langle \{p4, p9\}, p8 \rangle$ accounts for two of them, i.e., Figure 3(c) and (d), as there are two options for the selection of node "Stat". Readily, the total number of level-1 contexts corresponding to the direct

total precedence relations regarding $p8$ can be counted as $6 \times 3 = 18$.

3.3 Computation of Total Precedence

The third algorithm describes the procedure of constructing all the total precedence relations of depth $k + 1$ based on those of depth no more than k , with respect to a set P of productions, where $k \geq 1$. Readily, the set of total precedence relations of any depth can be recursively generated from the fundamental set of direct total precedence relations by using it.

Algorithm 3. Computation of total precedence relations.

Input. A set P of productions, \leq_p the direct total precedence relation on P , and the set Tpd of total precedence relations of depth no more than k , where $k \geq 1$.

Output. The set of total precedence relations of depth $k + 1$.

```

{
  Let  $Ptp$  and  $Tpr$  be one-dimensional arrays whose indexes range
  over  $P$ ;
  Let  $Tpd_k$  be the set of total precedence relations of depth  $k$ ;
  for each  $p \in P$  {
     $Ptp[p] = \{null\}$ ; // Initialize the elements of  $Ptp$ ;
  }
  for each  $ps \in Tpd$  {
    Let  $e = \langle P', p' \rangle$  be the root element of  $ps$ ;
     $Ptp[p'] = Ptp[p'] \cup \{ps\}$ ;
  }
  for each  $p \in P$  {
     $Ptp[p] = \emptyset$ ;
    for each  $e' = \langle P'', p' \rangle \in \leq_p$  {
      Let  $P'' = \{p_1, \dots, p_k\}$ ; //  $P''$  is a multiset;
       $Crt = Ptp(p_1) \times \dots \times Ptp(p_k)$ ;
       $E = \{e'\}$ ;
       $R = \emptyset$ ;
      for each  $(t_1, \dots, t_k) \in Crt$  such that
       $\exists i, j (t_i \neq null \wedge t_j \in Tpd_k)$  //  $1 \leq i \leq k$ ;
      Create a mapping  $f$  with domain  $P''$ ;
       $D = \emptyset$ ; //  $D$  is a multiset;
      for each  $t_i$  //  $1 \leq i \leq k$ ;
        if  $(t_i = null)$ 
           $f(p_i) = null$ ;
        else {
          Let  $t_i = (E_i, R_i)$ , and  $e_i$  be the root element;
           $f(p_i) = e_i$ ;
           $D = D \cup \{e_i\}$ ;
           $E = E \cup E_i$ ;
           $R = R \cup R_i$ ;
        }
      }
      Construct a linking relation  $r = (e', D, f)$ ;
       $R = R \cup \{r\}$ ;
       $Ptp[p''] = Ptp[p''] \cup \{(E, R)\}$ ;
    }
  }
}
return  $Tpr$ ;

```

The algorithm consists of two tasks. First, it partitions the set Tpd into $|P|$ distinct subsets in terms of the root node of the rooted tree that each total precedence relation (i.e., a precedence structure) forms, which comprise the

set Ptp .

Then, for any production p in P , it constructs the set of all the total precedence relations of depth $k + 1$ whose root elements share the same head p . More precisely, this set is the union of the subsets, each of which includes those relations whose root elements are a same direct total precedence relation with p as the head. Thus, in terms of each of these relations, say $\langle P'', p \rangle$, the algorithm constructs a corresponding subset of all the relations of depth $k + 1$ with $\langle P'', p \rangle$ as the root element.

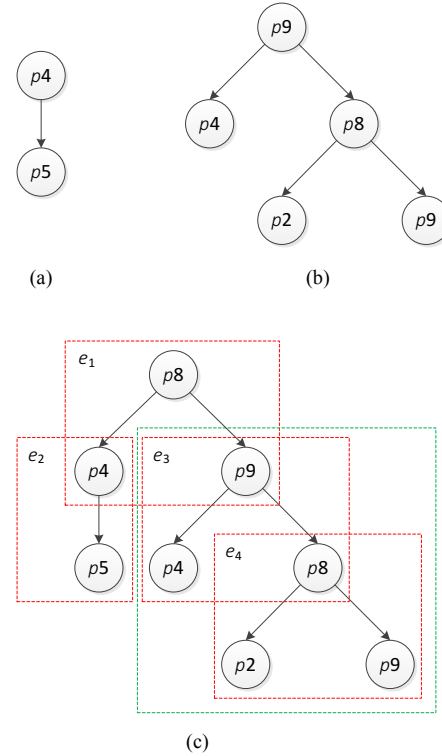


Fig. 4. The rooted trees of different depths that correspond to precedence structures.

To this end, it first conducts the Cartesian product of $|P''|$ selected elements from Ptp whose indexes exactly comprises the multiset P'' . Second, it screens the ordered tuples in the product to make sure that each chosen one can be utilized to generate a proper linking mapping from P'' to the union of $\{null\}$ and the set of root elements of its constituents, and that the resulting precedence structure must be of depth $k + 1$. The latter is guaranteed when one constituent of the tuple is of depth k . Third, for each chosen tuple, it generates the linking relation, which is composed of the head $\langle P'', p \rangle$, the set of tails, i.e., the root elements of the constituents of the tuple, and the newly created linking mapping, and then constructs the consequent precedence structure (E, R) , where E and R comprise all the involved direct total precedence relations and relevant linking relations, respectively.

A total precedence relation is commonly represented as a precedence structure. A precedence structure can visually form a rooted tree. In Figure 4, (a), (b), and (c) show three rooted trees that correspond to three precedence structures, called ps_1 , ps_2 , and ps_3 , respectively. Apparently, they are of depth 1, 2, and 3, respectively.

Example 2. A precedence structure $ps_3 := (E, R)$ on the production set P of the RGG in Figure 1, where:

- $E = \{e_1, e_2, e_3, e_4\}$, in which $e_1 = \{\{p4, p9\}, p8\}$, $e_2 = \{\{p5\}, p4\}$, $e_3 = \{\{p4, p8\}, p9\}$, $e_4 = \{\{p2, p9\}, p8\}$;
- $R = \{r_1, r_2\}$, in which $r_1 = (e_1, \{e_2, e_3\}, f_1)$ with $f_1(p4) = e_2$ and $f_1(p9) = e_3$, $r_2 = (e_3, \{e_4\}, f_2)$ with $f_2(p4) = \text{null}$ and $f_2(p8) = e_4$;
- $e_t = e_1$.

In the rooted tree corresponding to ps_3 , as depicted in Figure 4(c), the four fragments enclosed by red dashed rectangles are the elements e_1 , e_2 , e_3 , and e_4 , respectively, which are direct total precedence relations, i.e., precedence structures of depth 1.

The subtree enclosed by the green dashed rectangle corresponds to the precedence structure ps_2 .

Given the set of total precedence relations of depth no more than 2 including ps_1 and ps_2 (the set of direct total precedence relations involved) as input, the algorithm will generate a set of total precedence relations of depth no more than 3, where ps_3 is involved. That is, ps_3 is created on the basis of ps_1 and ps_2 , together with the direct total precedence relation e_1 .

3.4 Computation of Context

It is known that a total precedence relation, i.e., a precedence structure (E, R) , forms a rooted tree in such a way that each direct total precedence relation in E corresponds to a subtree of depth 1 and they are glued to each other in terms of the linking relations in R .

The fourth algorithm calculates all the level- i contexts of a production and respective extended productions in terms of a total precedence relation, i.e., a precedence structure, with respect to it. Readily, the diagram of a rooted tree corresponding to a precedence structure offers a more comprehensible perspective for the algorithm.

The algorithm is composed of two consecutive tasks. The first task is to transform a total precedence relation (a rooted tree) into a set of direct total precedence relations (rooted trees of depth 1).

To this end, the algorithm creates an empty set, adds the rooted tree to it, and repeats the subsequent four steps until all the elements in it become rooted trees of depth 1. First,

it randomly selects a rooted tree from the set and locates an outmost subtree of length 1 (with the root node, say p); then, it figures out all the possible context- i p 's according to the subtree, where $i \geq 1$; next, for each of those extended productions, say p' , it constructs a new tree from the original one by pruning the subtree except the root (this node is preserved since it is also a leaf of another subtree to which this one is linked via a linking relation in R) from it and substituting p' for the root p , and puts it into the set; and finally, it deletes the originally selected tree from the set. After that, the set includes only rooted trees of depth 1, any of which corresponds to a direct total precedence relation.

Algorithm 4. Computation of contexts and extended productions.

Input. A set P of productions, a total precedence relation $\langle M, p_0 \rangle = (E_0, R_0)$ of depth h .

Output. The level- h contexts of p_0 and corresponding extended productions.

```

{
  Cps = {(E0, R0)};
  while (∃ps ∈ Cps such that ps is not a direct total precedence
  relation) {
    Let ps = (E, R);
    Find an element e ∈ E such that e is not the head of any
    linking relation in R;
    Let e = ⟨P', p⟩ and P' = {p1, ..., pk}; // k ≥ 1;
    Let r = (e', D, f) ∈ R such that e ∈ D;
    Let e' = ⟨P'', p''⟩;
    Let Fun1[p] = h, Fun2[p] = l;
    Let l(P') = {Q1, ..., Qm}; // m ≥ 1;
    Cps = Cps \ {ps};
    for each (t1, ..., tk) ∈ h(Q1) ∪ ... ∪ h(Qm) {
      Construct a level-i context (U, Z), i ≥ 1;
      Construct a context-i p with (U, Z), called p';
      P''' = (P' \ {p}) ∪ {p'};
      e'' = ⟨P''', p''⟩;
      D' = D \ {e};
      Create a linking mapping f': P''' → D' ∪ {null} such that
      f'(p') = null, and for any other production p''' ∈ P''',
      f'(p''') = f(p''');
      r' = (e'', D', f');
      E' = (E \ {e, e'}) ∪ {e''};
      R' = R \ {r} ∪ {r'};
      Construct a precedence structure ps' = (E', R');
      Cps = Cps ∪ {ps'};
    }
    Cps = Cps \ {ps};
  }
  Cnt = ∅;
  Xdp = ∅;
  for each ps ∈ Cps {
    Construct a level-h context (U, Z);
    Cnt = Cnt ∪ {(U, Z)};
    Construct a context-h p0 with (U, Z), called q;
    Xdp = Xdp ∪ {q};
  }
  return (Cnt, Xdp);
}

```

In the second task, it constructs, for each element in the set, a level- h context and based on it, an accompanying extended production as well.

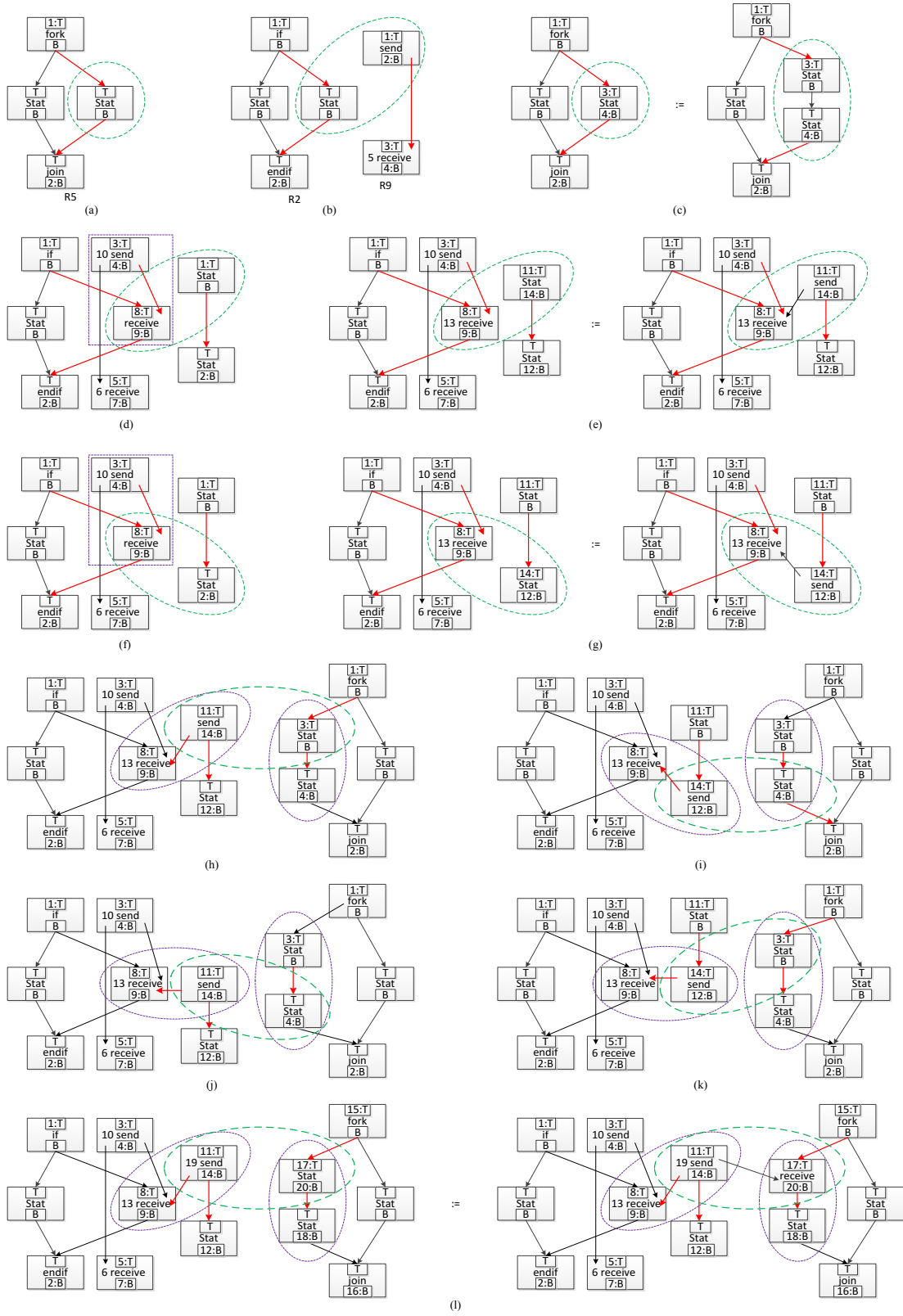


Fig. 5. Computation of Contexts. (a) A level-1 context of p_4 . (b) A level-1 context of p_8 . (c) A context-1 p_4 . (d) A level-2 context of p_9 . (e) A context-2 p_9 . (f) A level-2 context of p_9 . (g) A context-2 p_9 . (h-k) Four level-3 contexts of p_8 . (l) A context-3 p_8 .

Example 3. Computation of contexts.

Figure 5 depicts some of the contexts and context-equipped productions with respect to the RGG discussed above, which provides a visual demonstration of the computation process of Algorithm 4.

In each part of Figure 5, the subgraph enclosed by a green dashed ellipse is the left or right graph of some production, and the subgraph enclosed by a purple dashed rectangle or ellipse is the underlying production of a context-equipped production.

Figure 5(a) shows a level-1 context of p_4 , which is the output of Algorithm 4 when taking the direct total precedence relation ps_1 whose precedence structure is given in Figure 4(a) as input. The context-equipped p_4 , numbered as p_{10} , is shown in (c). It is also abbreviated to context-1 p_4 .

Figure 5(b) presents a level-1 context of p_8 , with respect to the direct total precedence of e_4 whose precedence structure is shown in Figure 4(c). The corresponding context-equipped p_8 (named by p_{11} above) is given in Figure 2(c). Taking the total precedence relation ps_2 (comprised of e_3 and e_4) shown in Figure 4(b) as input, Algorithm 4 generates two level-2 contexts of p_9 , as depicted in (d) and (f). That is, based on the right graphs of p_4 and p_{11} (whose underlying production is p_8), the algorithm produces all the level-2 contexts of p_9 . The quantity of contexts depends on the number of options when creating the left graph of the given production from other productions' right graphs. Accordingly, two context-equipped p_9 are illustrated in (e) and (g), and numbered as p_{12} and p_{13} , respectively.

The computation of the total precedence relation ps_3 is on the basis of ps_1 and ps_2 . In a similar way, when taking the total precedence relation ps_3 as input, Algorithm 4 generates four level-3 contexts of p_8 , as shown in Figure 5(h)-(k). That is, the contexts are produced based on the right graphs of p_{10} and p_{12} , or of p_{10} and p_{13} . A context-equipped p_8 that corresponds to the level-3 context in (h) is demonstrated in (l).

4. Complexity Analysis

In this section, the complexities of the proposed algorithms in the preceding section are analyzed.

Algorithm 1 calls a procedure $\text{FindRedex}(H, G)$ to generate the set of subgraphs of H that are redexes of the marked graph G . A procedure similar to the callee was proposed in [12] with time complexity $O(|H|^{|G|})$, where $|H|$ or $|G|$ denotes the number of nodes involved in it. Then, the time complexity of the algorithm directly follows:

Proposition 1. *The time complexity of Algorithm 1 is $O(m^2 n^2 r^r)$, where m is the number of productions in P , n is the maximal number of components in the left or right graphs of productions in P , and r is the maximal number of nodes in any of the components.*

Proposition 2. *The time complexity of Algorithm 2 is $O((mn)^{n+1} r^{rn})$, where m , n and r are as in Proposition 1.*

Proof. The algorithm mainly consists of a for-loop that nests other three sequential for-loops.

In the outmost loop, the first nested for-loop also nests another for-loop, which takes time $O(mn^2)$. The line next to it is the calculation of a k -ary Cartesian product over sets of redexes of components in $p.L$, each of which contains at most $mn \times r^r$ elements, where mn and r^r are the maximal number of elements in Cm_R and $M[C, C']$, respectively. Thus, the time complexity is $O((mnr^r)^n)$, since k equals to n in the worst case.

In the second nested for-loop, the first line takes $O(n)$, because the production to which each redex in the ordered tuple of Rlt corresponds is clearly indicated beforehand, according to the underlying assumption. Therefore, this for-loop takes time $O(n(mnr^r)^n)$.

Moreover, the time complexity of the last nested for-loop is at most $O(n(mnr^r)^n)$, for the cardinality of $Dtp[p]$ is no more than that of $Rlt[p]$.

In summary, the nested part of the outmost loop takes time $O(n(mnr^r)^n)$. Consequently, the time complexity of the whole procedure is $O((mn)^{n+1} r^{rn})$, which is the product of the time complexity of the outmost for-loop and that of its nested part. ■

Proposition 3. *The time complexity of Algorithm 3 is $O(m^{n+2} n l^n)$, where m and n are as in Proposition 1, and l is the cardinality of Tpd .*

Proof. The algorithm consists of three sequential for-loops. It is evident that the first two loops take $O(n)$ and $O(l)$, respectively.

The last one is a four-layer nested for-loops. According to the structure, its time complexity can be expressed as $O(d_1 d_2 d_3 d_4)$, where d_1 , d_2 , d_3 and d_4 are the maximal number of times traversed in the outmost, second, third, and inmost for-loop, respectively. We proceed inwards from outside of the structure.

First, d_1 is the number of productions in P , that is, $d_1 = m$. Next, d_2 equals to the cardinality of \leq_p , which is no more than $m \times m^n = m^{n+1}$. It is obvious that d_3 is actually the cardinality of the Cartesian product over k sets from Ptp , where k denotes the number of tails in a direct total precedence relation. Since Ptp is a partition of

Tpd , each element of the former must be a subset of the latter. Thus, $d_3 < l^n$. As for the inmost loop, d_4 readily equals to n , which is exactly the same as the exponent appearing in the preceding inequation.

Consequently, the last structure takes $O(m^{n+2}nl^n)$, i.e., the product of the complexities of the four constituents. Readily, it is also the time complexity of the algorithm. ■

Theorem 1. *The time complexity of Algorithm 4 is $O((n! r^{rn})^{2n^{h-1}})$, where n and r are as in Proposition 1, and h is the depth of the input total precedence relation.*

Proof. The main part of the algorithm is a two-layer structure: a for-loop nested within a while-loop, followed by another for-loop.

As to the former, the total number of times it is traversed is the product of that of the outmost while-loop and of the inmost for-loop. We consider the while-loop first. In the worst case, the input total precedence relation (E_0, R_0) corresponds to a complete n -ary rooted tree of depth h . Then, the cardinality of E_0 , i.e., the number of subtrees of depth 1 that compose it, can be expressed as:

$$1 + n + \dots + n^{h-1} = \frac{n^h - 1}{n - 1} \quad (1)$$

Suppose the number of times the inmost for-loop is traversed at the worst case be w . In each traversal of the while-loop, it takes a rooted tree out from Cps , and then puts as many as w revised ones that comprise one less subtrees back into it. This process is repeated until each element in Cps becomes a rooted tree of depth 1, i.e., it only involves a root element. Consequently, the maximal number of times the loop is traversed can be expressed as:

$$1 + w + w^2 \dots + w^{(\frac{n^h - 1}{n - 1} - 1)} = \frac{w^{\frac{n^h - 1}{n - 1}} - 1}{w - 1} \quad (2)$$

Further inference to Formula (2) can be done as follows:

$$\begin{aligned} \frac{w^{\frac{n^h - 1}{n - 1}} - 1}{w - 1} &< \frac{w}{w - 1} \cdot w^{\frac{n^h - 1}{n - 1} - 1} = \frac{w}{w - 1} \cdot w^{\frac{n(n^{h-1} - 1)}{n - 1}} \\ &< \frac{w}{w - 1} \cdot w^{2n^{h-1}} = O(w^{2n^{h-1}}) \end{aligned}$$

Note that w exactly equals to the number of extended context-1 productions that can be produced from a direct total precedence relation, by Definition 5. In the worst case, the relation consists of one head and n tails, and each component of the latter's right graphs contains r^r redexes of any component of the former's left graph. To be exact, of the loop variable (t_1, \dots, t_k) for the for-loop, each element t_i can be any of the r^r redexes of the corresponding component of the left graph in any component of any tail's right graph, where $1 \leq i \leq k$, and $k = n$. Thus, for each permutation of the tails, there are

$(r^r)^n = r^{rn}$ possible ordered tuples, where n is the maximal number of choices for selecting one component from each tail, and r^r is the maximal number of redexes in each component. Furthermore, the number of permutations for the tails is $n!$. Therefore, $w = n! r^{rn}$. Substituting the result for w in Formula (2) yields $(n! r^{rn})^{2n^{h-1}}$.

As to the latter, the number of times it is traversed is $n^{h-1} \times n = n^h$.

Consequently, the time complexity of the algorithm is $O((n! r^{rn})^{2n^{h-1}})$. ■

5. Discussion

5.1 Applicability of the Algorithms

From the perspective of time complexity, the above four algorithms seem rather complicated at first glance. Nevertheless, they are applicable in practical scenarios due to the following three causes.

First, the parameters that characterize a graph grammar are usually small constants, and cannot change in any computation. That is, the parameters are the nature of a graph grammar that will not vary with host graphs in parsing or derivation processes under different situations.

Second, the worst cases theoretically assumed in the analysis of the complexities can rarely happen in practice, and they are frequently quite small number in practical applications. Notice that the number of redexes with respect to a direct total precedence relation is surprisingly $n! r^{rn}$. However, this amount is merely a theoretical upper bound that accounts for all the possibly matched subgraphs, no matter in which situation all the redexes can simultaneously occur. An extreme situation in this case is a host graph where all the nodes in the host graph are labeled with the same symbol and the directed edges between them are completely connected. However, such host graphs can rarely be encountered in practice. As an example, consider the graph grammar depicted in Figure 1, the number of redexes with respect to a direct total precedence relation is theoretically $2! 4^{2 \times 4}$, whereas in the practical computation it is less than 10 in most cases.

Third, the contexts of a graph grammar can be achieved as the output from merely one execution of the algorithms, and then they can be repeatedly utilized in the process of derivation and parsing of this grammar at any time afterwards.

5.2 Application of Context

Context offers a concrete way for designers or users to grasp the meaning of an implicit graph grammar by directly observing the productions instead of enumerating

the members of the language. A context of a production characterizes a potential circumstance, under which it can be applied for derivation, a means usually employed to generate members of the language. Conversely, the context can also be regarded as a circumstance under which the production can be applied for parsing. Any production is self-explanatory for what it is for, whereas the contexts at distinct levels indicate at which situations it can be applied. These two aspects together clearly show the intension or meaning of a production, from the point view of derivation. Consequently, contexts can facilitate the comprehension of a graph grammar by synthesizing the meanings of its constituents so as to constitute the overall characteristics of the members of its language.

Moreover, context can facilitate the improvement of parsing performance. A general parsing algorithm is always a necessity for graph grammar formalisms. Backtracking is the main cause of high time complexity of a general parsing algorithm. In the process of parsing a host graph, when some unexpected (false positive) redexes are found and the corresponding reductions are conducted accordingly, then a final graph may be obtained that is not the initial graph of the involved graph grammar and to which no more reductions can be done. This situation gives rise to backtracking. A redex is called false positive if the situation in which the redex lies does not match any of the contexts of the production. Consequently, identifying the false positive redexes so as to avoid unexpected reductions is an effective way to improve parsing performance. Apparently, context matching can serve this purpose.

Noticeably, the proposed approach to context computation can be directly applied to practical graph grammars specifying real-world visual languages, e.g., BPMN (Business Process Model and Notation), ER diagrams, UML diagrams, WebML (Web Modeling Language), chemical diagrams, and so on, since these graph grammars are concrete examples of the underlying formalisms where the specification of nodes and edges in productions is entirely inherited from the formalisms.

6. Conclusion

On the basis of RGG, a representative of implicit context-sensitive graph grammar formalism, this paper has proposed an approach to the computation of context according to the formal definition of context, and presented the time complexities of the partially ordered algorithms involved. The method can facilitate the applicability of implicit graph grammars, as contexts of the productions are essential information for the comprehension of graph grammars and the improvement of parsing performance of parsing algorithms. Besides, the method can be generalized to other implicit context-

sensitive graph grammar formalisms without much effort.

In the future, further investigation will be conducted to explore more application scenarios of context, and a support system for context computation and visualization in a context-sensitive graph grammar framework will be developed as well.

Acknowledgments

This work is supported in part by the National Science Foundation of China under grants 61170089 and 91318301.

References

- [1] Chang S. K. (1987) "Visual Languages: A Tutorial and Survey". IEEE Software, 4(1), pp. 29–39.
- [2] Marriott K. (1994) "Constraint Multiset Grammars". IEEE Symposium on Visual Languages, St. Louis, Missouri, pp. 118–125.
- [3] Ferrucci F., Pacini G., Satta G., et al. (1996) "Symbol-Relation Grammars: A Formalism For Graphical Languages". Information and Computation, 131(1), pp. 1–46.
- [4] Chang S. K. (1971) "Picture Processing Grammar and Its Applications". Information Sciences, Vol. 3, pp.121–148.
- [5] Lakin F. (1987) "Visual Grammars for Visual Languages". 7th National Conference on Artificial Intelligence, pp. 683–688.
- [6] You K. C., Fu K. S. (1979) "A Syntactic Approach to Shape Recognition Using Attributed Grammars". IEEE Transactions on Systems, Man and Cybernetics, 9(6), pp. 334–345.
- [7] Costagliola G., Deufemia V., Polese G. (2007) "Visual Language Implementation Through Standard Compiler-Compiler Techniques". Journal of Visual Languages and Computing, 18(2), pp. 165–226.
- [8] Rozenberg G. (Ed.) (1997) "Handbook on Graph Grammars and Computing by Graph Transformation, Vol.1: Foundations". World Scientific.
- [9] Engels G., Kreowski H. J., Rozenberg G. (Eds.) (1999) "Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 2: Applications, Languages, and Tools". World Scientific.
- [10] Ehrig H., Kreowski H. J., Montanari U., Rozenberg G. (Eds.) (1999) "Handbook of Graph Grammars and Computing by Graph Transformation, Vol.3: Concurrency, Parallelism, and Distribution". World Scientific.
- [11] Rekers J., Schürr A. (1997) "Defining and Parsing Visual Languages with Layered Graph Grammars". Journal of Visual Languages and Computing, 8(1), pp. 27–55.
- [12] Zhang D., Zhang K., Cao J. (2001) "A Context-Sensitive Graph Grammar Formalism for the Specification of Visual Languages". The Computer Journal, 44(3), pp.187–200.
- [13] Nagl M. (1979) "A Tutorial and Bibliographical Survey on Graph Grammars". International Workshop on Graph Grammars and Their Application to Computer Science and Biology, Lecture Notes in Computer Science, Vol. 73, Springer Verlag, pp. 70–126.
- [14] Nagl M. (1987) "Set Theoretic Approaches to Graph Grammars". International Workshop on Graph Grammars and Their Application to Computer Science, Lecture Notes in Computer Science, Vol. 291, Springer Verlag, pp. 41–54.
- [15] Shi Z., Zeng X., Zou Y., et al. (2018) "A Temporal Graph Grammar Formalism," Journal of Visual Languages and Computing, Vol. 47, pp. 62–76.

- [16] Kong J., Zhang K., Zeng X. (2006) "Spatial Graph Grammars for Graphical User Interfaces". *ACM Transactions on Computer-Human Interaction*, 13(2), pp. 268–307.
- [17] Zhao C., Kong J., Zhang K. (2010) "Program Behavior Discovery and Verification: A Graph Grammar Approach". *IEEE Transactions on Software Engineering*, 36(3), pp. 431–448.
- [18] Roudaki A., Kong J., Zhang K. (2016) "Specification and Discovery of Web Patterns: A Graph Grammar Approach". *Information Sciences*, Vol. 328, 528-545.
- [19] Liu Y., Zeng X., Zou Y., Zhang K. (2018) "A Graph Grammar-Based Approach for Graph Layout," *Software: Practice and Experience*, 49(8), pp. 1523–1535.
- [20] Kong J., Barkol O., Bergman R., Pnueli A., Schein S., et al. (2012) "Web Interface Interpretation Using Graph Grammars". *IEEE Transactions on System, Man, and Cybernetics – Part C*, 42(4), pp. 590–602.
- [21] Chen L., Huang L., Chen L. (2015) "Breeze Graph grammar: A Graph Grammar Approach for Modeling the Software Architecture of Big Data-Oriented Software Systems". *Software: Practice and Experience*, 45(8), pp. 1023–1050.
- [22] Liu Y., Zhang K., Kong J., Zou Y., Zeng X. (2018) "Spatial Specification and Reasoning Using Grammars: From Theory to Application," *Spatial Cognition & Computation*, Taylor & Francis, 18(4), pp. 315–340.
- [23] Pfaltz J. L., Rosefeld A. (1969) "Web Grammars". *International Joint Conference on Artificial Intelligence*, pp. 609–619.
- [24] Zeng X., Han X., Zou Y. (2008) "An Edge-Based Context-Sensitive Graph Grammar Formalism". *Journal of Software*, 19(8), pp. 1893–1901. (in Chinese)
- [25] Liu Y., Shi Z., Wang Y. (2018) "An Edge-Based Graph Grammar Formalism and Its Support System". *International DMS Conference on Visualization and Visual Languages*, pp.101–108.
- [26] Zou Y., Zeng X., Han X. (2008) "Context-Attributed Graph Grammar Framework for Specifying Visual Languages". *Journal of Southeast University (English Edition)*, 24(4), pp. 455–461.
- [27] Bottoni P., Taentzer G., Schürr A. (2000) "Efficient Parsing of Visual Languages Based on Critical Pair Analysis and Contextual Layered Graph Transformation". *IEEE Symposium on Visual Languages*, pp. 59–60.
- [28] Zou Y., Lü J, Tao X. (2019) "Research on Context of Implicit Context-Sensitive Graph Grammars". *Journal of Computer Languages*, Vol. 51, pp. 241–260.

Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jvlc

CROSSIDE: A Design Space for Characterizing Cross-Surface Collaboration by Sketching

Jorge-Luis Pérez-Medina^{a,*}, Jean Vanderdonckt^{b,**} and Santiago Villarreal-Narvaez^b

^aIntelligent & Interactive Systems Lab (SI² Lab), Sede Queri, Av. De los Granados, Universidad de Las Américas (UDLA), Quito 170504, Ecuador

^bLouRIM Institute, Place des Doyens, 1, Université catholique de Louvain (UCL), B-1348 Louvain-la-Neuve, Belgium

ARTICLE INFO

Article History:

Submitted 4.8.2019

Revised 4.30.2019

Second Revision 5.20.2019

Accepted 8.15.2019

Keywords:

Design methodology

Displays

Gesture recognition

Graphical User Interfaces

Interactive systems

Pervasive computing

Ubiquitous computing

ABSTRACT

This paper introduces, motivates, defines, and exemplifies CROSSIDE, a design space for representing capabilities of a software for collaborative sketching in a cross-surface setting, i.e., when stakeholders are interacting with and across multiple interaction surfaces, ranging from low-end devices such as smartwatches, mobile phones to high-end devices like wall displays. By determining the greatest common denominator in terms of system properties between forty-one references, the design space is structured according to seven dimensions: user configurations, surface configurations, input interaction techniques, work methods, tangibility, and device configurations. This design space is aimed at satisfying three virtues: *descriptive* (i.e., the ability to systematically describe any particular work in cross-surface interaction by sketching), *comparative* (i.e., the ability to consistently compare two or more works belonging to this area), and *generative* (i.e., the ability to generate new ideas by identifying potentially interesting, under covered areas). A radar diagram graphically depicts the design space for these three virtues to enable a visual representation of one or more instances.

© 2019 KSI Research

1. Introduction

In many domains of human activity, across them [1] and within [2], *sketching* is largely used as a quick, efficient, and cost-effective tool for expressing ideas, illustrating design concepts and solutions as well as for sharing them with diverse people [3]. During the early stages [4] of product development until its final completion, sketching immediately plays some role when co-design is a must [5]. Sketching facilitates the sharing of abstract ideas and the insights inside a team while offering a common ground for the discussion, especially when the participants come from different cultural


and social backgrounds. Sketching usually uses basic drawings to foster everyone's participation [6].

Sketching, as one particular form of drawing, belongs to the first human intellectual skills and abilities that are acquired even before speaking, writing, and precise drawing. By one year, infants understand that a sequence of sounds form a word that symbolically represents an action, a relation, or an object they can point to rather than reaching it. Our human ability to perceive and to recognize elements in a graphical representation is in itself a statement of the expression power yielded by such form of expression.

A *sketch*, coming from the Greek word *σχεδιοζ* (*schedios - done ex tempore*), consists in a quick provisional free-hand drawing executed without any constraint in any medium. A sketch may convey a message, record or develop an abstraction or may be used as a mean for explaining something, for example an image. A sketch is considered as a very quick drawing aimed at communicating a message, which can be understood, misinterpreted, or even ignored. But there is always a message that is different from the one of a *drawing*. For these reasons, within many professions such as industrial and architecture design [5], representing actions, ob-

*Corresponding author

**Principal corresponding author

 jorge.perez.medina@udla.edu.ec (Jorge-Luis Pérez-Medina);

jean.vanderdonckt@uclouvain.be (Jean Vanderdonckt);

santiago.villarreal@uclouvain.be (Santiago Villarreal-Narvaez)

 http://investigacion.udla.edu.ec/udla_teams/jorge-perez/ (

Jorge-Luis Pérez-Medina); <https://www.uclouvain.be/jean.vanderdonckt/> (

Jean Vanderdonckt); <https://uclouvain.be/fr/santiago.villarreal/> (

Santiago Villarreal-Narvaez)

ORCID(s): 0000-0003-4864-0480 (Jorge-Luis Pérez-Medina);

0000-0003-3275-3333 (Jean Vanderdonckt); 0000-0001-7195-1637 (

Santiago Villarreal-Narvaez)

DOI reference number: 10-18293/JVLC2019-N1-016

jects, and their relations using different forms of sketches such as drafts, blueprints, and prototypes, has always been very important. They make visible the stakeholder's contribution, which could be different than expected.

The audience of stakeholders involved in sketching activities is now wider. Consequently, the pool of software tools to support collaborative sketching, both commercial and from the academia, also becomes more filled. *How much sketching activities could be supported by collaborative tools* is therefore a key question addressed in this paper. Researchers should be informed about capabilities of existing tools to update their research agenda and to better understand similarities and differences. Practitioners should also be informed about which tool would match their requirements for conducting collaborative sketching.

The remainder of this paper is structured as follows: Section 2 reviews sketching across disciplines involving some collaboration and discusses selected works; based on this, Section 3 introduces, motivates, and defines CROSSIDE, a design space for characterising capabilities of a software for collaborative sketching; Section 4 exemplifies some instantiations of this design space; Section 5 concludes this paper by explaining how this design space could be systematically used to describe, compare, and invent tools for collaborative sketching on multiple surfaces of interaction.

2. Related Work

In Information and Communications Technologies (ICT) as well as in computer and software systems engineering, a significant amount of resources is devoted to designing a concept, a service, a solution which could be later on revealed as inadequate. This could happen when the functional requirements are not satisfied, when the user experience is not met, or when the concept is simply not technologically feasible or too expensive. When such a mismatch is discovered late in the development life cycle or after the deployment, adapting what is required to be changed represents a high cost.

Early sketching in the development life cycle helps creating alternative solutions and comparing them. It helps to ensure that the right approach to design the right solution is put in place. It keeps the design and development right. With a sketch, designers and other stakeholders are able to identify figures, arrows, symbols and other elements that were deliberately chosen by a computer actor to communicate with stakeholders, to illustrate the requirements and share design ideas. It is more efficient and effective than any textual, graphical or formal specifications.

By using sketches, designers become more motivated, more creative, and perhaps more able to address the challenges of creating a successful design, and thus to produce a better design outcome [7]. By definition, prototypes are scaled-down versions of what will be built. Designers use them because they are faster and cheaper to create than the final blueprints. Sketching is a quick provisional drawing, it therefore matches the requirements of prototyping [3].

In Human-Computer Interaction (HCI) design, Collaborative

User-Centered Design (CUCD) process suggests using sketch to understanding and designing all the aspects of a user interface (UI) design and for getting involved a wide community of stakeholders, such as, but not limited to: user researcher, information architect, user interface interaction designers, user testing specialists, software developers, marketing personnel and software products leaders. Early sketches inform the development of User interface conceptual, interaction style, and even other related material such training resources, support services, online help, etc. UI usability can be seen as a design problem of “wicked nature”, as a problem to be solved – the more you try to solve the problem, the more you discover the complexity of the UI usability and more you are able to suggest solutions in the early design phase of the CUCD life cycle. This is because usually original usability problem has implications/consequences that cannot be known in advance. Sketches have been shown very powerful to build a consensus and a trade-off when usability problems are conflicting with other major quality factors, such as security. A UI sketch reveals how much a system could be usable, secure.

Sketching is the practice of drawing a rough outline or rough draft version of a final piece of art. Sketching is an aid to thought. Sketches are used as a mean of designing. Design by sketching has its foundations on the participatory design approach [8] in which a person not trained, qualified or experienced is an active and essential participant in the design process. As a communication tool, sketching can be used as a way of graphically specifying abstract ideas. It is a message from the designer to stakeholders. It can be understood by the receiver, misinterpreted, or ignored, but there is definitely a message. This message must be validated when there is a consensus to be achieved between the designer and someone else, for which designers often use limited or scaled versions of what is being designed.

A common ground to disciplines relying on is that stakeholders, particularly designers and end-users, whatever their background and skills in sketching and design are, feel actively engaged [9]. End-users help in materializing some requirements such as usability [10]. Designers then annotate original sketches introduced by end-users, add illustrations, and further develop them. They usually proceed by iteratively sketching a concept at different levels of abstraction [11]. Ambler [12] defines UI Prototyping as an iterative analytical technique in which end users are actively involved, namely by providing feedback since the early development stages and continuously afterwards.

Sketching covers many domains of human activity and inside these domains, there are several works exploiting some form sketching for one or many sub-activities such as for example: problem analysis in general [13], computer science [2] (e.g., user experience support [14, 15], user interface design, prototyping, and recognition [16, 17, 18, 19, 20, 21, 22], cross-device UI design [4], user-centered design in agile projects [23, 24], system walkthrough [25]), system development (e.g., QUILL [26] for model-based design of web applications), flexible modelling [27] (e.g., FlexiSketch [28, 29]

for model sketching), RAPIDO [30] for web API development, sketching UML models (e.g., TAHUTI for sketching UML Class diagrams [31] and SketchML for various UML diagrams [20]), distributed software design [32, 33], task modelling [34], notation creation [29]), computer-supported collaborative work [35] (e.g., stakeholders' meetings [36], collocated tables for meetings [37] and interactive design spaces [38]), product and service design (e.g., sketching in design [35], extreme designing [39], industrial design [5], shape-changing products [40]), public displays [41], learning (e.g., classroom design studio [42], teaching geometry [43]), ideation [44] and concept generation [7], knowledge design, capture, and sharing [1], design in any area of engineering [7] (e.g., knot diagramming [45]). From these references, we can observe that a significant amount of work has been devoted to using sketching as a way to support collaboration among stakeholders during the software development life cycle, starting from requirements engineering to detailed design. It is particularly useful for those stages involving some form of graphical representation of artifacts, whether they are informal (as a screen shot or wireframe) or formal (e.g., a UML model). Many techniques have been successfully reported for expressing sketch grammars [6], with the need to take into account the context of use (i.e., the user, the devices/platforms, and the environment) [46] to get context-aware sketching [43].

3. A Design Space for Cross-Surface Collaboration by Sketching

We determined the greatest common denominator in terms of properties between the 41 aforementioned references, independently of their domain, provided that collaborative sketching is involved to some extent. This identification resulted into CROSSIDE, a design space expressing collaborative sketching according seven dimensions represented clockwise in Fig. 1: user configurations, surface configurations, input interaction techniques, work methods, tangibility, layout, and device configurations. Each dimension is organized according to a progressive degree of sophistication: each step starts from the simplest value found in the literature until the most sophisticated degree.

We considered this representation as adequate to satisfy three virtues that are considered important to characterize interaction as a model [47]: *descriptive* (the design space should be able to describe any work on collaborative sketching based on these seven dimensions), *comparative* (the design space should be able to compare two or more works on collaborative sketching to identify their similarities and differences and foster consistency based on the previous description) and *generative* (once a comparison is performed, the design space should be able to identify uncovered areas and generate new and interesting ideas, configurations).

These seven dimensions are not intended to be completely independent of each other. Rather, they are aimed at serving these three virtues. Moreover, a radar chart can be effectively used to graphically represent the values along these

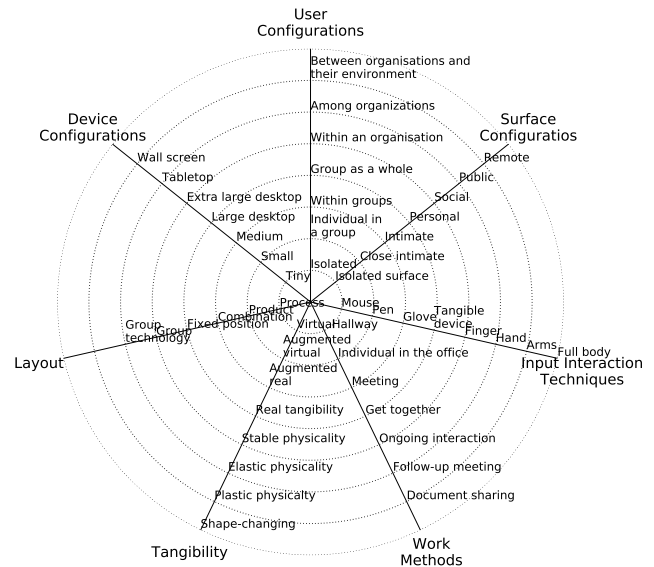


Figure 1: A design space for Cross-Surface Collaboration by Sketching.

dimensions by representing them on axes starting from the same origin. The steps on each dimension are not intended to be aligned or to be corresponding by concentric level. Therefore, steps joined on a same circle are not necessarily dependent, they only represent some progression.

3.1. User Configurations

The *user configurations* provide multiple ways to carry out a task by different stakeholders. Since many people could be located in different places, possibly at different levels of various organisations, the design space should consider the group configuration. Fig. 2a depicts various configurations among stakeholders involved in a distributed task [9]: *individual* (one task is carried out by one person in a group), *within groups* (one task is distributed across persons of a same group in the organisational structure), *group as a whole* (one task is carried out by one group of persons, independently of its internal organisation), *among groups* (one task is passed from one group to another), *within organisation* (one task is distributed across entities of the organisational structure), *among organisations* (when one task is distributed across several different organisations, all having their own internal structure), and *between organisations and their environment* (when tasks are exchanged between organisations and their common environment). Collaborative sketching requires stakeholders within groups because sketching tasks are distributed across people of a same group of the same organisational structure or not.

Fig. 3 graphically depicts an evolutive scale of the user configuration for different user configurations with respect to number of sessions and scenarios: an isolated stakeholder working alone, an individual stakeholder working as a group member, a single group of stakeholders, multiple groups of different stakeholders, multiple groups within the same organization (e.g., a group of representative end-users in a bank vs a group of interaction designers in the same bank), several

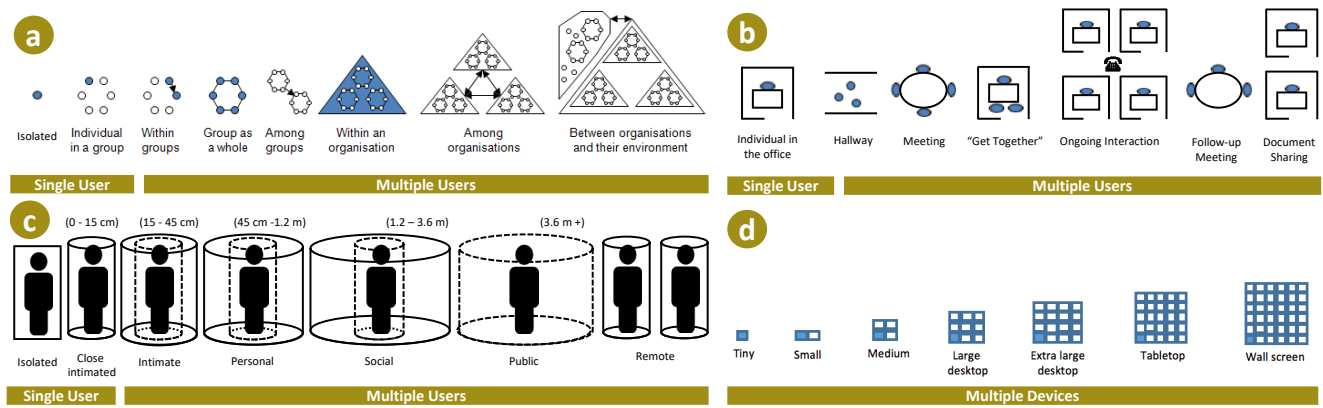


Figure 2: CROSSIDE dimensions: (a) the “user” dimension, (b) the “work method” dimension, (c) the “surface” dimension, and (d) the “device” dimension.

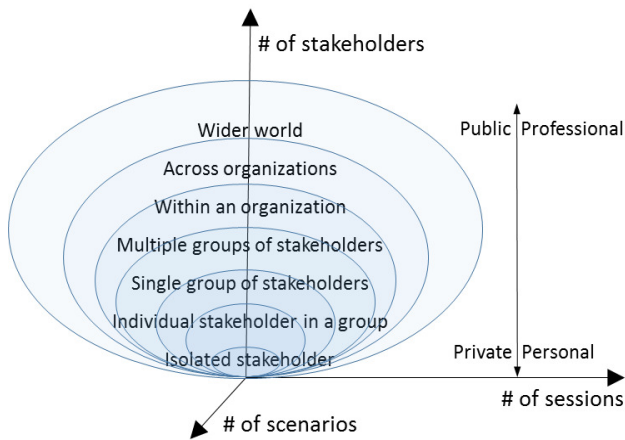


Figure 3: User configuration with sessions, stakeholders, and scenarios.

groups across organizations (e.g., an association of interaction designers in a particular domain of human activity) or the whole world (e.g., a community of practice could be developed that gathers people involved in a particular family of designs).

3.2. Surface Configurations

The *surface configurations* express a hypothetical distance between a stakeholder working alone and multiple ones collaborating in a personal, social, public or remote environment, thus touching the notion of territoriality. Fig. 2c differentiates configurations depending on concentric spatial zones centered around the stakeholder. The distance at which users feel comfortable interacting with others in tabletop environments depend on age and culture [37]. This study has contemplated the type of task or activity in which the users are engaged to influence tabletop design. In a completely isolated or close intimate space, a user works with personal interaction surfaces [48].

For example, a collaborative sketching system enables users testing UI prototypes on any surface size if the user is completely isolated. When users require a close intimate space, tiny or small surfaces will be used instead because

they facilitate face-to-face collaboration and eye-contact [5]. Stakeholders who are not designers or developers usually come to meetings with their own interaction surface (e.g., a smartphone, a tablet) and would like to see the UI prototype on their very right devices, even if they drew something on another, possibly larger, surface. Personal spaces offer more ample possibilities than intimate spaces because users sitting around a table or standing side-by-side can share work-spaces by using multiple displays, large, or extra large desktops, tabletops, and wall screens. The social space takes place in a business-based interaction. Stakeholders and final users could interacting in shared work-spaces by using large tabletop display or wall screens.

For example, a social space is adopted when some kind of multiple user’s interactions in taking place in a space between 1.2–3.6 m. Public space allows the user to interact with a sizable audience. In this context, the users make use of multiple display. Small surfaces allow manipulation of the work-space, while large surfaces serve to display the information to the entire audience. Informational wall surfaces, like public displays [41], provide shared views to the users standing far away from the display. Interactive walls enable users to walk up to the display and interleave interaction and discussion among participants. Finally, in remote spaces, users share work-spaces with same or different times. A collaborative sketching system could offer most surface configurations. Stakeholders interact simultaneously with any device. In a collaborative session, users can sketch scenes in a private way or separated into small groups, perform a task privately and then communicate the results to all stakeholders [11]. They can interact using integrated environments composed of horizontal or vertical displays regardless of the dimensions of these devices.

3.3. Input Interaction Techniques

The *input interaction techniques* express the sophistication degree with which sketching is supported, ranging from an indirect manipulation to full direct manipulation. This dimension takes into account the use of materials and the naturalness of the interaction. A classical indirect pointing device is the “mouse” because its locus of control (i.e., the

physical space in which actions occur) is different from or outside the locus of application (i.e., where the actions are applied). “Pen-based” interaction enables the user to interact with the device by using a passive or active stylus rather than a mouse. While pen devices initially supported indirect manipulation, they now support direct manipulation, namely by touch technology, regardless the fingers or the pointers involved. The “glove” considers the family of lightweight and stretchable devices that combines hand posture sensing and tactile pressure. Manipulation is direct, by device intermediation, like with smartwatches, armbands, and rings. “Tangible” devices naturally offer direct manipulation by connecting objects and surfaces to digital information. Tangible UIs typically work on tabletop surface and embed the tracking mechanism inside or outside the surface [49]. The “finger” considers situations where the end user interacts directly with the surface by finger tracking. This dimension could be extended to using the whole hand, but requires technologies for hand pose recognition. The last step, i.e. “full body”, involves full-body gesture recognition, but its benefits for sketching are yet to be demonstrated.

3.4. Work Methods

The *work methods* characterise how interaction surfaces are spatially arranged according to a territoriality in the environment and how these environments are connected together (e.g., through Wi-Fi, LAN, WAN). Interaction surfaces could be *tiled*, *coupled*, *uncoupled*, or *positioned side-by-side* [48]. This dimension subsumes the physical location of each surface and how it is positioned (i.e., vertically, horizontally, in an oblique way). The dimension also considers the devices configurations of the five categories of technologies for collocated collaborative work classified by Wang et al. [5] namely: horizontal displays, large vertical displays, multiple displays, tangible interfaces, and integrated environments. Fig. 2b depicts typical setups of single and multiple users. Users can work alone in their own organisations. Multiple users work in corporate environments [9]: *hallway* (when an environment consists of any informal place where users could meet), *individual in the office* (when an environment only accommodates one user at a time, although this user can change over time), *meeting* (when the environment accommodates several users at a time for conducting a meeting), *get together* (when the environment accommodates several users for collaboration in general), *ongoing interaction* (when different environments involve many different users).

3.5. Tangibility

The *tangibility* dimension expresses to what extent the materialization of collaboration space is digital, physical, or mixed, ranging from applications in digital environments to the use of shape-changing devices. Digital environments define most classical applications where UI can be 1D (based on lines), 2D (based on surfaces), 2D1/2 (based on a space projected onto a surface) or 3D (in space). 1D UIs are typically based on command lines or instructions. In digital

environments, users interact with a minimal immersion degree. Information is presented as a stream of characters in a text terminal. 2D UIs are incorporated in an environment based on conventional bi-dimensional representations where the interaction techniques are supported by events. Their widgets and their composition enable creating complex interfaces. The 2D1/2 and 3D interfaces extend classic GUIs with the notion of overlap and depth perceptions.

Virtual describes real environments simulated by a computer where users are involved with a high immersion degree. Users interact with virtual objects in an infinite 3D space. On the opposite of the dimension, applications are developed in real environments and use of mechanisms to increase the perception of the user. *Augmented Virtual* are applications including virtual worlds generated by a computer. These applications incorporate virtual reality to replace the physical world and the virtual world predominates over the real. *Augmented Virtual* extends the physical reality perceived by incorporating virtual objects into the physical world, thus increasing the degree of immersion. In contrast, *Augmented Real* considers an otherwise real environment augmented by means of virtual objects [50]. In the *Augmented Real*, virtual objects increase the real world, the dominant medium.

Tangible UIs give physical form to digital information, employing physical artifacts both as representations and controls for computational media [51]. This is further refined into four steps depending how tangible objects are materialized: real tangibility, stable physicality, elastic physicality, and plastic physicality. Real tangibility attempts the reproduce the physical behavior of a real world object into the tangible object, or a sub-set of it. Stable physicality occurs when tangible objects never change their behavior, whether they are expected to mimic some real work or not. Elastic physicality groups all environments that use both physical materials with computational analysis and simulation. These environments are used to understand and represent the world. Plastic physicality occurs in the area of shape-changing UI, where the devices are artifacts whose surface and/or volume can be articulated and modulated with their spatial domain [40]. In physics and materials science, the physicality property is one form of adaptation, which could be decomposed into three major properties: *plasticity*, which describes the deformation of a material undergoing (non-reversible) changes of shape in response to applied forces; *elasticity*, which is the tendency of solid materials to return to their original shape after being deformed; and *viscosity*, which expresses to what extent a material is resistant with respect to deformation. Solid objects will deform when forces are applied on them; if the object is elastic, it will return to its initial shape and size when these forces are released. These two terms can be used for tangibility.

3.6. Layout

The *Layout* dimension refers to the facility layout concept [52] borrowed from Production and Operations Management aimed at optimizing the physical arrangement of

resources (e.g., a machine, a device, an operator) to maximize the quality and the quantity of the output, while minimizing the cost of involved resources. The different types of layout are [52, 53]:

- *Process* layout, when all resources performing similar type of operations are grouped at one location according to their functions. In our case, this means that all human, software and hardware resources required for each phase are gathered in the same place to support the collaboration, such as all resources for design in one place while development occurs in another place. The flow paths of information from one functional area to another vary from product to product. Usually the paths are long with backtracking possible.
- *Product* layout, when all resources are located according to the processing sequence of the product. In our case, resources are gathered in one location at a time depending on the phase and the locations are arranged in a sequence that follows the product life cycle, such as requirement, early analysis and design, advanced design, development, deployment, evaluation.
- *Combination* layout, when input to be processed come in different types and sizes. The combination layout is process layout where resources are arranged in a sequence to produce various types and sizes of products. The sequence of phases remains the same for the products having different types (e.g., only designing a concept vs sketching and designing) and sizes (e.g., sketching the home page of a web site or sketching the whole web site).
- *Fixed position* layout, when major physical resources remain in a fixed location (e.g., devices and platforms for sketching) and human resources are sent to this location depending on the phase. For example, a tabletop setup for collaborative sketching is heavy to move from one location to another and complex to re-calibrate, therefore stakeholders are brought to this location to ensure the phase. On the other hand, a small mobile sketching station could move from one location to another.
- *Group* layout, when product and process layout are combined in a particular layout called cell that satisfy a predefined set of requirements.
- *Group Technology layout*, when a group layout emerges from so-called Group Technology [53], which is aimed at analysing and comparing items to group them into families with similar characteristics. Families of inputs sharing similar requirements are grouped into cells, each cell being capable of satisfying all the requirements assigned to it. For example, phases sharing similar user requirements, even from different projects, will be conducted in the same place with the same resources.

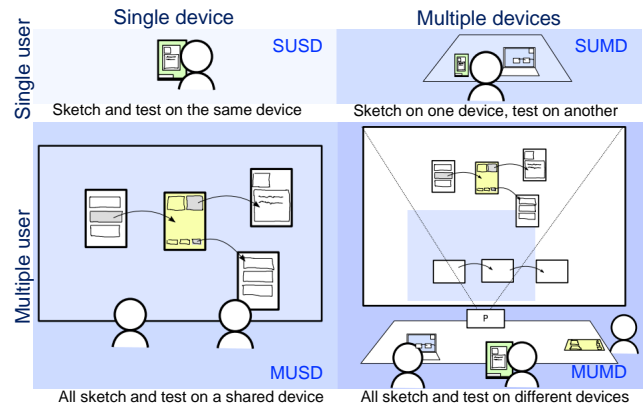


Figure 4: Quadrant of the four configurations.



Figure 5: Example of a SUSD configuration: one user with one device among several.

3.7. Device Configurations

The *device configurations* refer to the surface size of device independently of their density and orientation. Fig. 2d depicts the different dimensions for the device configurations. The sizes are categorized by generalised dimensions ranging from tiny to wall screen. A system could run on a variety of devices offering different sizes, performing scaling and resizing to accommodate variations induced by different screens. Four typical configurations based on stakeholders (or users) and their surfaces therefore emerge (Fig. 4):

1. *Single user-Single device (SUSD)*: a single stakeholder, such as an end-user, is working in isolation on one device only to sketch a UI (see figure 4 - top left). This device could be the very right device on which the final UI should run or another one. Therefore, this configuration is appropriate for conducting the sketching, prototyping, and testing activities as defined.
2. *Single user-Multiple devices (SUMD)*: a single stakeholder is sketching on multiple devices either simultaneously or asynchronously in order to assemble the various sketches into a coherent design scenario (see figure 4 - top right). Therefore, this configuration is appropriate for conducting the sketching, prototyping, and sharing/testing activities.

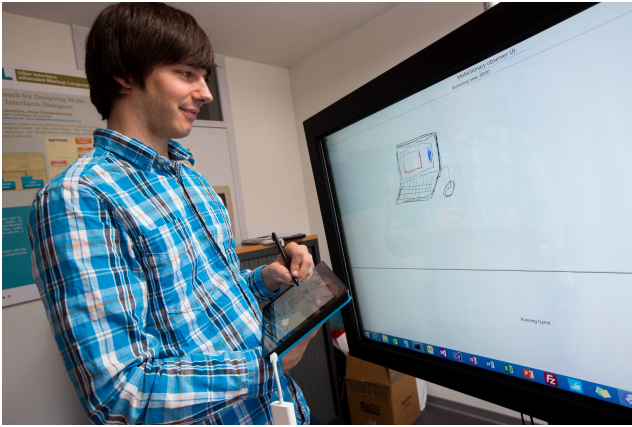


Figure 6: Example of a SUMD configuration: one user sketching on one device (a tablet) and testing on another (a wall screen).

3. *Multiple user-Single Device (MUSD)*: several stakeholders share a same device in order to interact together and to come up with an agreement on some design questions (see figure 4 - bottom left). This configuration is expected to mimic the classical whiteboard configuration where different stakeholders are together in front of a whiteboard and sketching things all together. Therefore, this configuration is appropriate for supporting the sharing/testing and discussing/reflecting activities, while it could be also used for the sketching and prototyping activities, but not primarily.
4. *Multiple users-Multiple devices (MUMD)*: several stakeholders exploit private and public devices to apply any decided modification on the design scenario (see figure 4 - bottom right). They could perform these actions first on their own private device (e.g., their tablet), and then propagate modifications to the whole scenario (displayed on a wall screen). They straightforwardly interact on the public device to collect immediate feedback from other stakeholders. Therefore, this configuration is appropriate for supporting the sharing/testing and discussing/reflecting activities.

3.8. Building process of the Design Space

The design space has been built by identifying the greatest common denominator of software features described from forty-one references in various domains. The process followed to build the design space was basically a middle-out approach that combines two sub-processes, while trying to satisfy the principle of separation of concerns regarding the three main aspects of a context of use (e.g., users and their interactive tasks, their platforms and devices, their physical and organisational environments [46]):

1. A *bottom-up approach* consisting of browsing each reference at a time, identifying any high-level factor supporting collaborative sketching by separating them for users, platforms, and environments. All values found for each factor were collected in the same set.

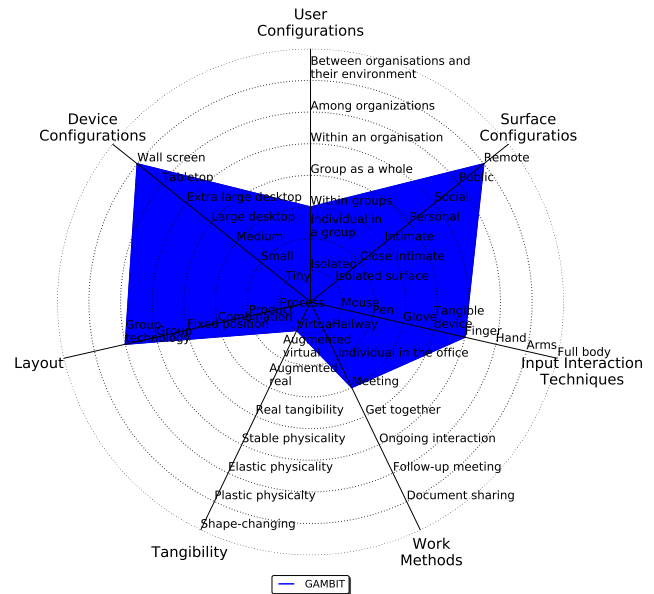


Figure 7: CROSSIDE instantiation for GAMBIT.

2. A *top-down approach* consisting of determining such high-level factors by looking after a theoretical definition in the literature.

For example, the “User configurations” factor was identified among several references, therefore encouraging us to find a taxonomy that is relevant enough to express values found for this factor. We discovered a similar concept in Mandviwalla and Olfman [54]. To make it suitable for our purpose, we expanded the possible values from values collected in the references and we adapted the definition accordingly to make it categorical. Fig. 2a graphically represents the resulting factor presented as a progressive dimension.

Another example is related to the “Surface configurations” factor. By extracting from each use case found in every reference, we wanted to categorise the surface projected on the ground that delineates the interaction space. When only one person is involved in a sketching, there are various comfort zones from social psychology. When several persons are involved in a collaborative sketching, the way they interact with their system depends on what they want to do with their sketch. The sketching activity itself is more frequently found in small surfaces, while sharing and discussing the sketching is more frequently found in medium to large surfaces. The scope of the input and the output determines the surface.

4. Instantiation of the Design Space

This section first instantiates CROSSIDE on GAMBIT, one of our systems for collaborative sketching, to check how the descriptive virtue is addressed. It then performs the instantiations for the external tools: BELONGINGS [55], SKETCHML [20], FLEXISKETCH [28, 29, 56], CALICO [57] and EVE [22]. For the instantiations of each of the tools, we superimpose the first drawing to address the comparative virtue.

At the end, all instances are presented in the same design space allowing to see their similarities and differences.

4.1. The CROSSIDE instantiation for GAMBIT

GAMBIT¹ (Gatherings And Meetings with Beamers and Interactive Tables) [11] is a freely accessible² multi-platform HTML5 environment for collaborative design by sketching for UI design, but not only, by supporting (Fig. 7):

- **Multi-fidelity:** it should enable stakeholders to provide material at any level of fidelity and to easily switch from one level of fidelity to another.
- **Multi-format:** while sketching remains at the heart of the service, it should accommodate various formats of data both as input and as output.
- **Multi-session:** it should offer the possibility to create, edit, one or many sessions simultaneously.
- **Multi-scenario:** within a single session, it should be able to manage several alternate scenarios at once, namely by representing explicitly the different scenarios and by linking them. For instance fork and join mechanisms could be used to map scenarios together and create different design paths for idea writing. Different stakeholders make explicit more design ideas so as to generate new design paths that were remained under explored before. This is very important to avoid the “Do not fall in love of your own design” observation: once a design alternative is selected as an official one, some stakeholders tend to keep it (they fall in love of their design) and to spend remaining time in refining this design as opposed to exploring alternate designs.
- **Multi-stakeholder:** a same session could be created to involve a wide community of stakeholders whose roles and responsibilities may evolve over time. When the role or the responsibility of a stakeholder changes depending on the context of the problem, the tool should accommodate these changes flexibly.
- **Multi-access:** each stakeholder should contribute to a design session with her own devices. End-users as well as designers tend to create smaller sketches on small screen devices and larger sketches when the interaction surface increases, without necessarily adding more details. Multi-device is equally important to support cross-device interaction, e.g., when a stakeholder may switch from one device to another while participating in the same session, but with different roles.

¹The word “gambit” is used in chess in reference to a movement in which the player sacrifices a piece, usually a pawn, in exchange to some future advantage; in user interface design the gambit would be made by the designer who sacrifices sketches in exchange to know in advance how the interface would function in the hands of the end-users, allowing her to learn from the eventual observation of this usage.

²Accessible at <http://gambitsketch.appspot.com/>

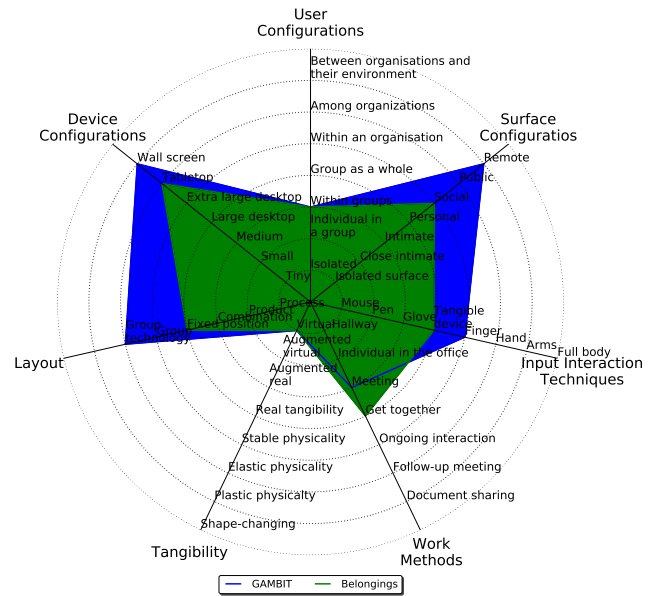


Figure 8: CROSSIDE instantiation for BELONGINGS superimposed to the instantiation for GAMBIT.

4.2. The CROSSIDE instantiation for BELONGINGS

BELONGINGS [55] consists of an interactive tangible tabletop surrounded by multiple interactive surfaces and smaller devices for collaborative query by sketching. The table was installed in a museum and designed to communicate indigenous traditional knowledge and cultural values. Fig. 8 depicts the design space instantiated for this environment superimposed on the previous one (Fig. 7) to compare the coverage of both tools. BELONGINGS can be operated in a fixed tabletop by two users working together in a social context. Unlike GAMBIT, BELONGINGS supports the use of tangible devices, thus being better with that respect on the dimension. Both software share a participatory design [8] approach as part of the development process: designers and final users work together to produce the final application.

4.3. The CROSSIDE instantiation for FLEXISKETCH

FLEXISKETCH [28, 29, 56] comes in two versions: the first uses tablets as sketching media and supports inexpensive, mobile sketching at any time and in any place, while the second version, known as “FlexiSketch Desktop”, runs on electronic whiteboards and provides a wide screen for collocated meetings. Multiple tablets can be connected to the desktop version over Wi-Fi, thus enabling end users to collaborate and simultaneously work in the same workspace regardless the multiple screens they are using. FLEXISKETCH enables participants to edit the workspace simultaneously, offers shared and private views, and allows group work and/or facilitating individual work. Fig. 9 depicts how FlexiSketch is instantiated on the design space. As a way for comparison, the instantiation of Gambit is also superimposed. FlexiSketch and Gambit have the similarities of considering the participatory design as part of the development process. Both tools

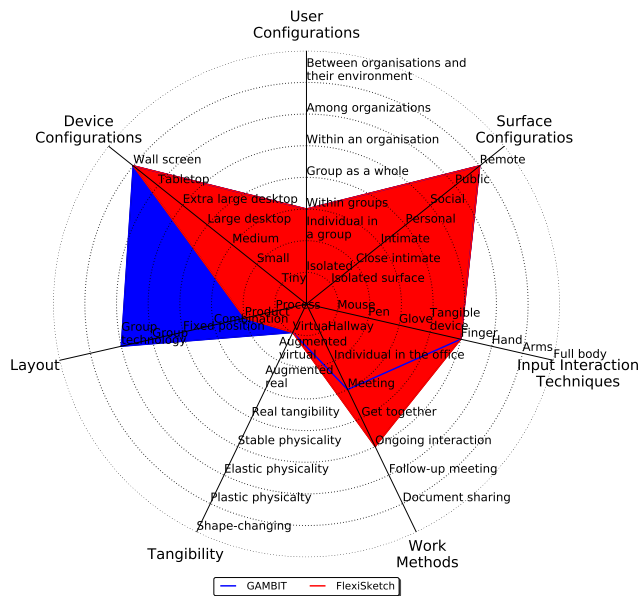


Figure 9: CROSSIDE instantiation for FLEXISKETCH on GAMBIT.

allow multiple concurrent accesses through the simultaneous use of multiple devices.

4.4. The CROSSIDE instantiation for CALICO

CALICO [57] is a free hand rapid design tool supporting early software design activities to be used with touch screen interfaces, such as interactive whiteboards and tablet PCs. CALICO enables designers creating designs on multiple canvases based on a client-server architecture, supporting up to 20 simultaneously active users [58]. A CALICO client is portable, supporting computers connected to electronic whiteboards, laptops, and tablets. Thus, CALICO supports collaborative work across multiple devices, allowing multiple designers to work synchronously on the same canvas or asynchronously on different canvases. This allows designers working in a group to branch off to their own canvas. Fig. 10 combines the instantiations of CALICO and GAMBIT.

4.5. The CROSSIDE instantiation for SKETCHML

SKETCHML [20] is a framework that offers the ability to define and recognize every kind of 2D graphical library, by using freehand drawing, to be used in the construction of user interfaces. The framework uses an empirical language based on XML. It language allows the compatibility of SketchML with other applications and services through various devices. Fig. 11 combines the instantiations of SKETCHML and GAMBIT.

4.6. The CROSSIDE instantiation for EVE

EVE [22] is a Sketch-based prototyping workbench that facilitates end-users to define their design through a set of low-fidelity sketches. The Low-fidelity representations are recognized and translated in medium fidelity representations, as well as in high fidelity prototypes. End-users realize the representations in a canvas of two dimensions. End-users

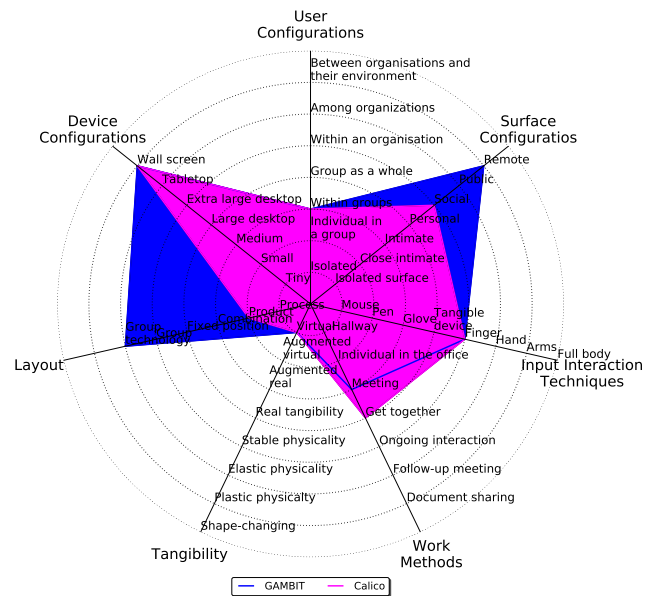


Figure 10: CROSSIDE instantiation for CALICO superimposed to the instantiation for GAMBIT.

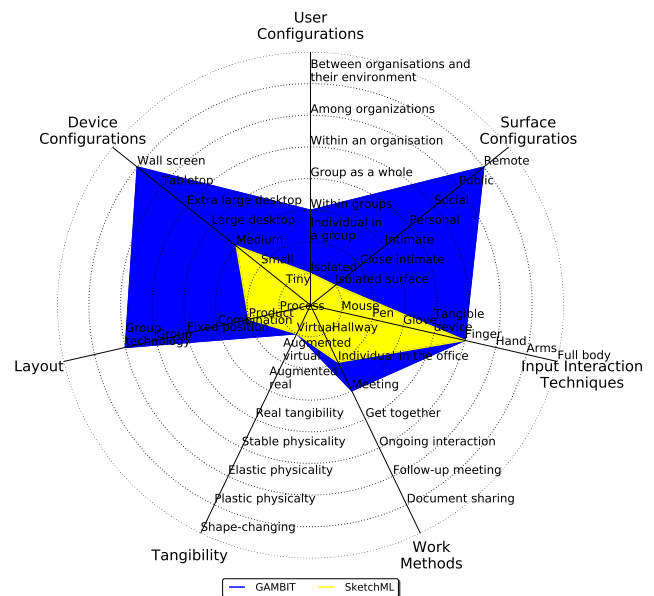


Figure 11: CROSSIDE instantiation for SKETCHML superimposed to the instantiation for GAMBIT.

can navigate through the three levels of loyalty. At each level it is possible to make the desired changes. For each of the fidelity, end-users can operate three modes. The design functionality, the configuration of the interaction and the preview of the prototype. Fig. 13 combines the instantiations of EVE and GAMBIT.

4.7. The comparative virtue of CROSSIDE

Fig. 14 combines all the instantiations of tools studied insofar in a radar diagram to facilitate the visual comparison of the tools (provided that they are not too numerous), such as the similarities and differences. The goal is to satisfy



Figure 12: Meeting work method for GAMBIT.

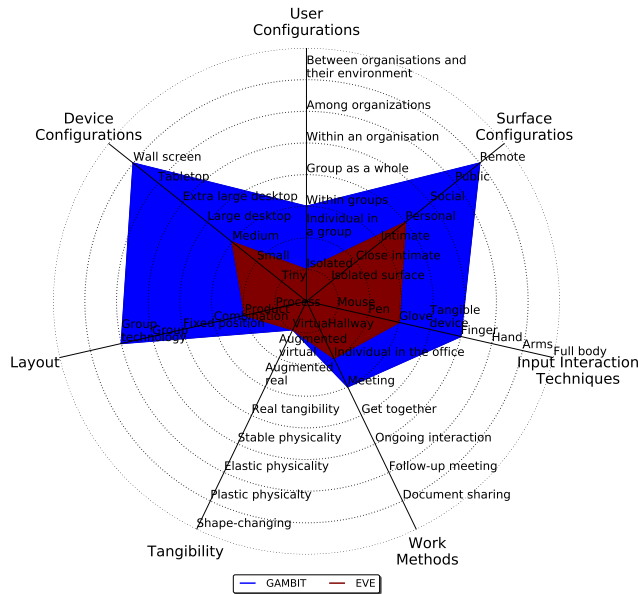


Figure 13: CROSSIDE instantiation for EVE superimposed to the instantiation for GAMBIT.

the comparative virtue of the design space, not to promote an ideal tool being the best along all dimensions. Rather, some tools are more advanced along some dimensions, say the “Device configurations”, while others are targeting more flexibility for other dimensions, like “User configurations”. For example, the range of “Device configurations” is well covered by both GAMBIT and CALICO, up to the three first steps of “Work methods”. Fig. 12 illustrates the “Meeting” work method, where two designers share a tabletop for sketching a prototype by pen that is rendered in real-time on the smartphone of an end-user. Sketching is in direct manipulation: the designer is sketching directly on the tabletop and manipulating the sketch in the same way.

4.8. The generative virtue of CROSSIDE

A close examination of Fig. 14 leads to the observation that some dimensions are pretty well covered, like “Device configurations” and “Surface configurations”, some others are moderately covered like “Layout”, “Work methods”, and “Input interaction techniques” while the remaining ones are

limited, such as “User configurations” and “Tangibility”. This suggests that these dimensions are welcome to be explored in further activities of a research agenda.

Regarding the “User configurations” dimension, we observe that the maximum step is “Within groups” because existing software do support multiple users (as represented in Fig. 4), but do not explicitly record and maintain the organisational structure among stakeholders. Their roles remain undifferentiated and the organisation structure is absent. A possible extension along this dimension would incorporate the explicit definition of such a structure, along with the roles played by stakeholders, especially in teams distributed in time and space. In this way, various types of organizations could be included, such as design teams, development companies like off-shore companies.

Regarding the “Device configurations” dimension, we observe that this dimension is the best one covered in the design space since the ultimate step is reached. Indeed, most software accommodate various types of devices and platforms, usually in a multi-platform or cross-device fashion.

Regarding the “Input Interaction techniques” dimension, the maximum step reaches finger-based interaction when drawing or sketching is conducted based on the physical movements of fingers, usually represented as uni- or multi-stroke gestures, thus limiting the interaction to 2D. 3D interaction is not really exploited, unless it is for the purpose of 3D objects. One could imagine for instance full-body gesture interaction to enable stakeholders to arrange pages of a web site dynamically in front of a wall-screen instead of moving them by point and click.

Regarding the “Work methods” dimension, the typical method observe seems to be “Ongoing interaction”, which means that interaction capabilities remain opportunistic and constant whatever the phase is and whoever the stakeholders are. Some software support design history with do, undo, redo (e.g., by replaying the sketching actions), but this activity is not synchronized with a software management tool or with a document sharing system to store the current status of a design. However, most software includes a facility to export its contents to be integrated in the software documentation.

Regarding the “Tangibility” dimension, we notice that this is the most limited dimension in all tools examined so far: most of them represent digital solutions where sketching is achieved on a 2D surface with limited beautification performed in this space. There are probably other tools for collaborative sketching in 3D exhibiting the capability to virtually augment the real world, but they do not belong to our initial list of references. For the practitioners, this means that no tool is available today to fulfill these needs. The design of interest remains also only digital. Although some software exist that address the needs of physical-digital interfaces or objects, such as so-called *phygital objects*, they are not integrated with collaborative design tools.

Regarding the “Layout” dimension, it is difficult to assess this dimension since the physical setup and arrangement of interaction surfaces and their users is not explicitly repre-

sented, as in a design topology. Most interaction surfaces are mobile or partially mobile, thus enabling them to be rearranged depending on the requirements of the phase. But there is no explicit mechanism to represent the requirements of a design that would be turned automatically into a physical configuration to support it. Instead, stakeholders change the layout themselves depending on the constraints they perceive, apart from fixed or heavy interaction surfaces that are only found in dedicated locations. An interesting extension here would be to explicitly consider the notion of *territoriality* [59] to express the public, private, and common spaces for collaborative actions that would be then transformed into an adaptable layout of interaction surfaces.

Regarding the “Device configurations” dimension, the covered part shows again that a wide range of device is typically supported, ranging from small to very large devices. This is especially the case when such devices benefit from a HTML5-compliant browser that enables them to communicate easily.

The design space obtained so far only reflects some significant dimensions identified among a set of forty-one references considered as representative instances of collaborative sketching in various domains of human activity, ranging from learning to industrial design. While this set of references covers several domains, we do not argue that its coverage is complete or representative enough of the vast majority of tools of interest. Therefore, our next step consists of conducting a Systematic Literature Review (SLR) [60] for identifying references relevant to collaborative sketching for multiple purposes:

1. To expand the coverage of reviewed works from our 41 selected references to a larger panel.
2. To address the reproducibility of the procedure for guaranteeing the coverage of the design space.
3. To address explicitly the generative virtue by discussing the most promising configurations on this design space which may serve for a research agenda in the near future.

For the moment, we can superimpose the instantiations of the design space performed for the 41 references. On one hand, this superimposition enables us to identify portions of each dimension that are more or less frequently covered, or not covered at all. But this analysis considers only one dimension at a time, which may be considered as reductive. On the other hand, the superimposition also enables us to identify the configurations that are the most or the least frequently adopted by tools. This does not mean that they are appropriate or not, but simply the coverage could be discussed. We prefer to perform this analysis on a set of references resulting from the SLR instead of our initial set.

5. Conclusion and Future Work

We presented CROSSIDE, a design space for representing capabilities of a software for collaborative sketching in a cross-surface setting. This design space consists of seven

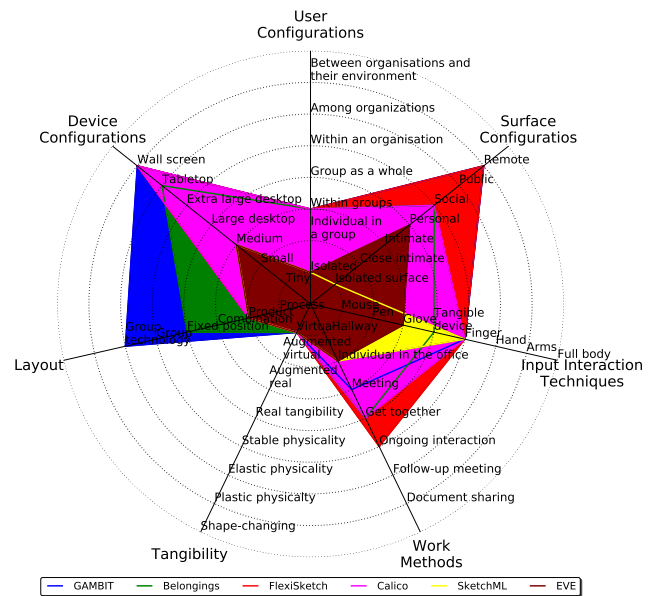


Figure 14: CROSSIDE instantiations for BELONGINGS, FLEXISKETCH, CALICO, SKETCHML, EVE and GAMBIT all combined at once.

dimensions (i.e., user configurations, surface configurations, input interaction techniques, work methods, tangibility, layout, and device configurations) resulting from a comparative analysis of 41 references in the domain. Each instantiation of these 41 references on the design space is graphically depicted as a radar diagram, which visually supports three virtues: descriptive, comparative, and generative.

Acknowledgments

The authors thank the anonymous reviewers for their constructive and patient comments on earlier versions of this manuscript.

References

- [1] S. McCrickard, Making Claims: The Claim as a Knowledge Design, Capture, and Sharing Tool in HCI, Morgan & Claypool, June 2012. URL: <https://www.morganclaypool.com/doi/abs/10.2200/S00423ED1V01Y201205HCI015>. doi:10.2200/S00423ED1V01Y201205HCI015, Synthesis Lectures on Human-Centered Informatics.
- [2] C. Gonzalez-Perez, Filling the Voids - From Requirements to Deployment with OPEN/Metis, in: Proc. of the Fifth Int. Conf. on Software and Data Technologies, Volume 1, ICISOFT'10, SciTePress, 2010.
- [3] R. van der Lugt, Functions of sketching in design idea generation meetings, in: Proc. of the 4th Conf. on Creativity & Cognition, C&C '02, ACM, New York, USA, 2002, pp. 72–79. URL: <http://doi.acm.org/10.1145/581710.581723>. doi:10.1145/581710.581723.
- [4] J. Lin, J. A. Landay, Employing patterns and layers for early-stage design and prototyping of cross-device user interfaces, in: Proc. of the SIGCHI Conference on Human Factors in Computing Systems, CHI '08, ACM, New York, NY, USA, 2008, pp. 1313–1322. doi:10.1145/1357054.1357260.
- [5] H. Wang, E. Blevis, Concepts that support collocated collaborative work inspired by the specific context of industrial designers, in: Proc. of the ACM Conf. on Computer Supported Cooperative Work, CSCW

- '04, ACM, New York, USA, 2004, pp. 546–549. URL: <http://doi.acm.org/10.1145/1031607.1031698>. doi:10.1145/1031607.1031698.
- [6] G. Costagliola, V. Deufemia, M. Risi, Sketch Grammars: A Formalism for Describing and Recognizing Diagrammatic Sketch Languages, in: Proc. of Eighth Int. Conf. on Document Analysis and Recognition, 29 August - 1 September 2005, ICDAR' 05, 2005, pp. 1226–1231. URL: <https://doi.org/10.1109/ICDAR.2005.218>. doi:10.1109/ICDAR.2005.218.
- [7] M. C. Yang, Observations on concept generation and sketching in engineering design, Research in Engineering Design 20 (2009) 1–11.
- [8] M. J. Muller, S. Kuhn, Participatory Design, Communications of the ACM 36 (1993) 24–28.
- [9] E. Berglund, M. Bång, Requirements for distributed user interface in ubiquitous computing networks, in: Proc. of Conf. on Mobile and Ubiquitous Multimedia, MUM '02, ACM Press, New York, USA, 2002.
- [10] J. Vanderdonckt, A. Beirekdar, Automated web evaluation by guideline review, J. Web Eng. 4 (2005) 102–117.
- [11] U. B. Sangiorgi, F. Beuvs, J. Vanderdonckt, User interface design by collaborative sketching, in: Proc. of the ACM Int. Conf. on Designing Interactive Systems, DIS '12, ACM, New York, NY, USA, 2012, pp. 378–387. URL: <http://doi.acm.org/10.1145/2317956.2318013>. doi:10.1145/2317956.2318013.
- [12] S. W. Ambler, 2007, Agile adoption rate survey results: March 2007, URL: <http://www.ambysoft.com/surveys/agileMarch2007.html>.
- [13] P. Sachse, W. Hacker, S. Leinert, External thought-does sketching assist problem analysis?, Applied Cognitive Psychology 18 (2004) 415–425.
- [14] B. Buxton, Sketching User Experiences: Getting the Design Right and the Right Design, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.
- [15] S. Greenberg, S. Carpendale, N. Marquardt, B. Buxton, Sketching User Experiences - The Workbook, Academic Press, 2012. URL: <http://store.elsevier.com/product.jsp?isbn=9780123819598>.
- [16] J. A. Landay, B. A. Myers, Interactive Sketching for the Early Stages of User Interface Design, in: Proc. of the ACM Int. Conf. on Human Factors in Computing Systems, CHI '95, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1995, pp. 43–50. URL: <http://dx.doi.org/10.1145/223904.223910>. doi:10.1145/223904.223910.
- [17] A. Coyette, S. Faulkner, M. Kolp, Q. Limbourg, J. Vanderdonckt, SketchiXML: Towards a Multi-agent Design Tool for Sketching User Interfaces Based on UsiXML, in: Proc. of 3rd Int. Conf. on Task Models and Diagrams, TAMODIA '04, ACM, NY, USA, 2004, pp. 75–82. URL: <http://doi.acm.org/10.1145/1045446.1045461>. doi:10.1145/1045446.1045461.
- [18] A. Coyette, S. Kieffer, J. Vanderdonckt, Multi-fidelity prototyping of user interfaces, in: M. C. C. Baranauskas, P. A. Palanque, J. Abascal, S. D. J. Barbosa (Eds.), Proc. of 11th IFIP TC13 Int. Conf. on Human-Computer Interaction, September 10–14, 2007, volume 4662 of INTERACT '07, Springer, 2007, pp. 150–164. URL: http://dx.doi.org/10.1007/978-3-540-74796-3_16. doi:10.1007/978-3-540-74796-3_16.
- [19] Z. Obrenovic, J.-B. Martens, Sketching Interactive Systems with Sketchify, ACM Transactions on Computer-Human Interaction 18 (2011) 1–38.
- [20] D. Avola, A. Del Buono, G. Gianforme, S. Paolozzi, R. Wang, SketchML: a Representation Language for Novel Sketch Recognition Approach, in: Proc. of the 2nd Int. Conf. on Pervasive Technologies Related to Assistive Environments, PETRA '09, ACM, New York, NY, USA, 2009, pp. 31:1–31:8. URL: <http://doi.acm.org/10.1145/1579114.1579145>. doi:10.1145/1579114.1579145.
- [21] D. Avola, L. Cinque, G. Placidi, SketchSPORE: A Sketch Based Domain Separation and Recognition System for Interactive Interfaces, in: A. Petrosino (Ed.), Image Analysis and Processing, ICIAP' 13, Springer, Berlin, Heidelberg, 2013, pp. 181–190.
- [22] S. Suleri, V. P. Sermuga Pandian, S. Shishkovets, M. Jarke, Eve: A sketch-based software prototyping workbench, in: Extended Abstracts of the ACM Int. Conf. on Human Factors in Computing Systems, CHI EA '19, ACM, New York, NY, USA, 2019, pp. LBW1410:1–LBW1410:6. URL: <http://doi.acm.org/10.1145/3290607.3312994>. doi:10.1145/3290607.3312994.
- [23] P. McInerney, F. Maurer, UCD in agile projects: dream team or odd couple?, Interactions 12 (2005) 19–23.
- [24] T. Buchmann, Towards tool support for agile modeling: Sketching equals modeling, in: Proceedings of the 2012 Extreme Modeling Workshop, XM '12, ACM, New York, NY, USA, 2012, pp. 9–14. doi:10.1145/2467307.2467310.
- [25] C. Lewis, P. G. Polson, C. Wharton, J. Rieman, Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces, in: Proc. of the ACM Conf. on Human Factors in Computing Systems, CHI '90, ACM, New York, USA, 1990, pp. 235–242. doi:10.1145/97243.97279.
- [26] V. Genaro Motti, D. Raggett, S. Van Cauwelaert, J. Vanderdonckt, Simplifying the development of cross-platform web user interfaces by collaborative model-based design, in: Proceedings of the 31st ACM International Conference on Design of Communication, SIGDOC '13, ACM, New York, NY, USA, 2013, pp. 55–64. URL: <http://doi.acm.org/10.1145/2507065.2507067>. doi:10.1145/2507065.2507067.
- [27] H. Ossher, A. van der Hoek, M.-A. Storey, J. Grundy, R. Bellamy, Flexible modeling tools (flexitools2010), in: Proc. of 32nd ACM/IEEE Int. Conf. on Software Engineering-Volume 2, ICSE '10, ACM Press, New York, USA, 2010, pp. 441–442.
- [28] D. Wüest, N. Seyff, M. Glinz, FlexiSketch: A Mobile Sketching Tool for Software Modeling, in: D. Uhler, K. Mehta, J. Wong (Eds.), Proc. of Mobile Computing, Applications, and Services, volume 110 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Springer, 2013, pp. 225–244. doi:10.1007/978-3-642-36632-1_13.
- [29] D. Wüest, N. Seyff, M. Glinz, Flexisketch team: Collaborative sketching and notation creation on the fly, in: Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE Int. Conf. on, volume 2, 2015, pp. 685–688. doi:10.1109/ICSE.2015.223.
- [30] R. Mitra, Rapido: A Sketching Tool for Web API Designers, in: Proc. of 24th Int. Conference on World Wide Web, WWW '15 Companion, Int. World Wide Web Conf. Steering Committee, Republic and Canton of Geneva, Switzerland, 2015, pp. 1509–1514. doi:10.1145/2740908.2743040.
- [31] T. Hammond, R. Davis, Tahuti: A geometrical sketch recognition system for uml class diagrams, in: AAAI Spring Symposium on Sketch Understanding, 2002, pp. 59–68.
- [32] N. Mangano, M. Dempsey, N. Lopez, A. van der Hoek, A demonstration of a distributed software design sketching tool, in: Proc. of the 33rd Int. Conf. on Software Engineering, ICSE '11, ACM, New York, NY, USA, 2011, pp. 1028–1030. URL: <http://doi.acm.org/10.1145/1985793.1985985>. doi:10.1145/1985793.1985985.
- [33] J. Melchior, J. Vanderdonckt, P. Van Roy, A model-based approach for distributed user interfaces, in: Proc. of the 3rd ACM Symposium on Engineering Interactive Computing Systems, EICS '11, ACM, New York, NY, USA, 2011, pp. 11–20. URL: <http://doi.acm.org/10.1145/1996461.1996488>. doi:10.1145/1996461.1996488.
- [34] J.-L. Pérez-Medina, S. Dupuy-Chessa, A. Front, A Survey of Model Driven Engineering Tools for User Interface Design, in: M. Winckler, H. Johnson, P. Palanque (Eds.), Proc. of 6th Int. Workshop on Task Models and Diagrams for User Interface Design, TAMODIA '07, volume 4849 of Lecture Notes in Computer Science, Springer, Berlin, 2007, pp. 84–97. doi:10.1007/978-3-540-77222-4_8.
- [35] G. Johnson, M. D. Gross, J. Hong, E. Y. Do, Computational support for sketching in design: A review, Foundations and Trends in Human-Computer Interaction 2 (2009) 1–93.
- [36] M. Johansson, M. Arvola, A case study of how user interface sketches, scenarios and computer prototypes structure stakeholder meetings, in: Proc. of the 21st British HCI Group Annual Conference on People and Computers, vol. 1, BCS-HCI '07, British Computer Society, Swinton, UK, 2007, pp. 177–184. URL: <http://dl.acm.org/citation.cfm?id=1531294.1531318>. doi:10.1145/1531294.1531318.

- [37] J. R. Wallace, S. D. Scott, Contextual design considerations for co-located, collaborative tables, in: Proc. of the 3rd IEEE Int. Workshop on Horizontal Interactive Human Computer Systems, TABLETOP'08, IEEE, 2008, pp. 57–64.
- [38] J. Bowen, A. Dittmar, A Semi-formal Framework for Describing Interaction Design Spaces, in: Proc. of the 8th ACM Symposium on Engineering Interactive Computing Systems, EICS '16, ACM, New York, NY, USA, 2016, pp. 229–238. URL: <http://doi.acm.org/10.1145/2933242.2933247>. doi:10.1145/2933242.2933247.
- [39] B. S. da Silva, V. C. O. Aureliano, S. D. J. Barbosa, Extreme designing: binding sketching to an interaction model in a streamlined HCI design approach, in: Proc. of 7th Brazilian symposium on human factors in CS, ACM Press, New York, USA, 2006, pp. 101–109.
- [40] M. K. Rasmussen, G. M. Troiano, M. G. Petersen, J. G. Simonsen, K. Hornbæk, Sketching shape-changing interfaces: Exploring vocabulary, metaphors use, and affordances, in: Proc. of the ACM Int. Conf. on Human Factors in Computing Systems, CHI '16, ACM, New York, NY, USA, 2016, pp. 2740–2751. URL: <http://doi.acm.org/10.1145/2858036.2858183>. doi:10.1145/2858036.2858183.
- [41] J. Müller, F. Alt, D. Michelis, A. Schmidt, Requirements and design space for interactive public displays, in: Proc. of the 18th ACM Int. Conf. on Multimedia, MM '10, ACM, New York, NY, USA, 2010, pp. 1285–1294. URL: <http://doi.acm.org/10.1145/1873951.1874203>. doi:10.1145/1873951.1874203.
- [42] D. Loksa, N. Mangano, T. D. LaToza, A. v. d. Hoek, Enabling a classroom design studio with a collaborative sketch design tool, in: Proc. of the Int. Conf. on Software Engineering, ICSE '13, IEEE Press, Piscataway, NJ, USA, 2013, pp. 1073–1082. URL: <http://dl.acm.org/citation.cfm?id=2486788.2486935>.
- [43] G. Costagliola, M. De Rosa, V. Fuccella, Local context-based recognition of sketched diagrams, Journal of Visual Languages and Computing 25 (2014) 955–962.
- [44] R. Bellamy, M. Desmond, J. Martino, P. Matchen, H. Ossher, J. Richards, C. Swart, Sketching tools for ideation (nietrack), in: 33rd Int. Conference on Software Engineering, ICSE '11, ACM, New York, NY, USA, 2011, pp. 808–811. doi:10.1145/1985793.1985909.
- [45] M. D. Rosa, A. Fish, V. Fuccella, R. Saleh, S. Swartwood, G. Costagliola, A toolkit for knot diagram sketching, encoding and re-generation, in: G. Polese, V. Deufemia (Eds.), The 22nd International Conference on Distributed Multimedia Systems, DMS 2016, Salerno, Italy, November 25–26, 2016., KSI Research Inc. / Knowledge Systems Institute Graduate School, 2016, pp. 16–25. URL: <https://doi.org/10.18293/DMS2016-035>. doi:10.18293/DMS2016-035.
- [46] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, J. Vanderdonckt, A Unifying Reference Framework for multi-target user interfaces, Interacting with Computers 15 (2003) 289–308.
- [47] M. Beaudouin-Lafon, Designing interaction, not interfaces, in: M. F. Costabile (Ed.), Proc. of the ACM Int. Working Conference on Advanced Visual Interfaces, AVI '04, ACM Press, 2004, pp. 15–22. URL: <https://doi.org/10.1145/989863.989865>. doi:10.1145/989863.989865.
- [48] J. Coutaz, C. Lachenal, S. Dupuy-Chessa, Ontology for multi-surface interaction, in: Proc. of the IFIP TC13 Int. Conf. on Human-Computer Interaction, 1st-5th September 2003, INTERACT '03, IOS Press, 2003.
- [49] O. Shaer, E. Hornecker, Tangible user interfaces: past, present, and future directions, Foundations and Trends in Human-Computer Interaction 3 (2010) 1–137.
- [50] P. Milgram, F. Kishino, A taxonomy of mixed reality visual displays, IEICE Transactions on Information and Systems 77 (1994) 1321–1329.
- [51] B. Ullmer, H. Ishii, Emerging frameworks for tangible user interfaces, IBM Systems Journal 39 (2000) 915–931.
- [52] B. J. Finch, Operations Now: Profitability, Processes, Performance, McGraw-Hill/Irwin, Boston, 2006.
- [53] J. Wu, J.-y. Lv, Z.-k. Ye, M. Rui, Optimization Design of Facilities Layout in a Certain Machining Shop, in: Proc. of 2nd Int. Conf. on Information Technology and Management Engineering, ITME'17, DEStech Publications, Inc, Lancaster, Pennsylvania, 2017. URL: <http://dpi-proceedings.com/index.php/dtcse/article/view/7986>. doi:10.12783/dtcse/itme2017/7986.
- [54] M. Mandviwalla, L. Olfman, What Do Groups Need? A Proposed Set of Generic Groupware Requirements, ACM Transactions on Computer-Human Interaction 1 (1994) 245–268.
- [55] R. Muntean, Considering Collaboration in ?elewkw—Belonging, in: Proc. of the 3rd Int. Workshop on Interacting with Multi-Device ecologies" in the wild", Cross-Surface'16, 2016. URL: http://cross-surface.com/papers/Cross-Surface_2016-2_paper_9.pdf.
- [56] D. Wüest, N. Seyff, M. Glinz, Collaborative sketching and notation creation with FlexiSketch Team, Software Engineering 2017 (2017).
- [57] N. Mangano, A. Baker, A. Van Der Hoek, Calico: a prototype sketching tool for modeling in early design, in: Proc. of the 2008 Int. Workshop on Models in software engineering, ACM Press, New York, USA, 2008, pp. 63–68.
- [58] N. Mangano, T. D. LaToza, M. Petre, A. van der Hoek, Supporting informal design with interactive whiteboards, in: Proc. of the ACM Int. Conf. on Human Factors in Computing Systems, CHI '14, ACM, New York, NY, USA, 2014, pp. 331–340. URL: <http://doi.acm.org/10.1145/2556288.2557411>. doi:10.1145/2556288.2557411.
- [59] J. Thom-Santelli, D. Cosley, G. Gay, What do you know?: Experts, novices and territoriality in collaborative systems, in: Proc. of the ACM Int. Conf. on Human Factors in Computing Systems, CHI '10, ACM, New York, NY, USA, 2010, pp. 1685–1694. URL: <http://doi.acm.org/10.1145/1753326.1753578>. doi:10.1145/1753326.1753578.
- [60] B. Kitchenham, R. Pretorius, D. Budgen, O. P. Brereton, M. Turner, M. Niazi, S. Linkman, Systematic literature reviews in software engineering – a tertiary study, Information and Software Technology 52 (2010) 792 – 805.

Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jvlc/

Collaborative E-learning Environments with Cognitive Computing and Big Data

Mauro Coccoli^a, Paolo Maresca^b, Andrea Molinari^c

^aUniversity of Genoa, DIBRIS, Genoa, Italy

^bFederico II University, DIETI, Naples, Italy

^cUniversity of Trento, DII, Trento, Italy, and School of Industrial Engineering and Management, Lappeenranta University of Technology, Finland

ARTICLE INFO

Article History:

Submitted 4.8.2019

Revised 4.30.2019

Second Revision 5.20.2019

Accepted 8.13.2019

Keywords:

Cognitive computing

Big data

Collaborative systems

Technology-enhanced learning

E-learning.

ABSTRACT

The actual scenario of e-learning environments and techniques is fast-changing from both the technology side and the users' perspective. In this vein, applications and services as well as methodologies are evolving rapidly, running after the more recent innovations and thus adopting distributed cloud architectures to provide the most advanced solutions. In this situation, two influential technological factors emerge: the former is cognitive computing, which can provide learners and teachers with innovative services enhancing the whole learning process, also introducing improvements in human-machine interactions; the latter is a new wave of big data derived from heterogeneous sources, which impacts on educational tasks and acts as enabler for the development of new analytics-based models, for both management activities and education tasks. Concurrently, from the side of learning techniques, these phenomena are revamping collaborative models so that we should talk about communities rather than classrooms. In these circumstances, it seems that current Learning Management Systems (LMS) may need a redesign. In this respect, the paper outlines the evolutionary trends of Technology-Enhanced Learning (TEL) environments and presents the results achieved within two experiences carried on in two Italian universities.

© 2019 KSI Research

1. Introduction

Technology and society are undergoing a continuous evolution process, which is pushing innovation and driving the development of novel solutions in almost any field of application. The wide adoption of such solutions in a plethora of services that are commonly available on the Web for daily operations heavily impacts on users' behaviors and expectations. Specifically, with reference to the Technology-Enhanced Learning (TEL) scenario, we observe significant steps ahead in techniques and methodology. As a consequence, technological solutions are subjected to continuous upgrades to cope with these, to the aim of improving the quality of services, the

usability, the overall performances, the effectiveness of education, and to provide learners with a more pervasive experience. Accordingly, e-learning environments and the relevant tools have been growing in complexity, i.e., traditional Learning Management Systems (LMS), based on a centralized architecture, are moving towards clusters of Massive Open Online Courses (MOOC) platforms in cloud-based distributed architectures [1] and Learning Objects (LO) include more and more videos and multimedia interactive artifacts. Concurrently, LMSs have been exploring new spaces of possibilities; among these, we mention the adoption of users' profiling techniques and analytics to the aim of tailoring personalized learning paths and activities as well as to predict students' careers, to achieve a further empowerment of the individual learning model. A clear synthesis of the more significant evolutionary steps of learning solutions is presented in [2] where the changes

^aCorresponding author

Email address: mauro.coccoli@unige.it

ORCID: 0000-0001-5802-138X

occurred in tools, services, learning strategies, methodologies, and delivery techniques are arranged along a timeline. It is worth noticing that the achieved progresses are not just replacements of the previously adopted solutions, yet they are characterized by one or more of the following: (i) extension of capabilities; (ii) improvement of performances; (iii) promotion of different educational approaches and methodologies; (iv) change in the way contents are delivered; (v) change in the users' interaction model. Despite the methodology is sound, the analysis concerns the period from 2001 to 2015 and, since then, many changes have occurred, mainly due to the rise of big data and cloud computing.

In this respect, we highlight that the introduction of cognitive computing can enable a number of new functionalities [3] and that the adoption of this new paradigm also reflects on the learning process, as schools and universities must face with new jobs and new training demands since new knowledge and skills should be promptly delivered [4], also pushed by big data. As a result, it seems that LMSs have reached their maturity stage in the innovation adoption curve, since they are pervasively adopted by learning providers at any level, from the primary schools up to higher education institutions, to the aim of guaranteeing the most effective implementation of TEL solutions. Nevertheless, most LMSs hardly can cope with the high degree of interoperability and complexity that novel paradigms require, hence they should expand their boundaries offering new services and rethink the whole design process, considering the labor market and industry needs too. In this respect, the paper discusses the evolutionary trends of e-learning and presents the results achieved within two experiences carried on in two Italian universities: (i) an innovative didactic approach to software engineering, adopting advanced technologies and mixed solutions in a cloud infrastructure; (ii) a novel collaborative learning environment that surpasses the typical functionalities of the more prominent LMSs.

We highlight that such experiences are driving TEL platforms towards new standards where both learners and teachers, rather than technology, will be at the center.

The remainder of the paper is structured as follows: in Section 2 we present some related works investigating these topics; Section 3 outlines the main drivers for the development of the next generation TEL platforms; in Section 4 we report the experience of a cognitive-computing-based laboratory, while in Section 5 we depict a newly developed e-learning collaborative environment. Section 6 concludes the paper also giving hints on future development directions.

2. Related works

As highlighted in the previous section, we are going to focus on two phenomena: (a) big data, and (b) cognitive computing. In fact, the possibility of gathering information from a wide number of heterogeneous sources, combined with the unprecedented opportunities of processing such data by means of sophisticated cognitive computing techniques, is reshaping the technological scenario of e-learning applications, allowing the improvement of existing methodologies as well as imagining new ones.

Moreover, the possibility of recording, storing and aggregating information, can significantly improve learning performances of both students and teachers [5]. For these topics, we report a number of related works that illustrate different solutions showing how variedly these two aspects can empower e-learning methodologies and how platforms are evolving with new functionalities. From a quick analysis of the relevant literature, i.e., searching indexing services such as Science Direct, Scopus and Web of Science, we found several publications on these challenging topics, enlightening that, more and more, LMSs offer such services among basic tools or as add-on functionalities.

2.1 The impact of big data on e-learning

Due to the presence of big data, a number of significant changes has occurred in education and in e-learning systems. Their impact on the academic environment is investigated in [6], where the authors' expectation is that a change will suddenly occur in the way e-learning is approached by both students and teachers. In this respect, based on the currently available software solutions, they propose a system architecture fostering universities to constitute consortia to analyze, organize and access huge data sets in common, in a cloud-based environment. A similar approach is also considered in [7], which proposes a strategy to improve outcomes of the education process through the collaboration among universities at an international level, to the aim of supporting the teaching and learning process by means of shared e-learning services. The integrated framework presented links the individual impact with the organizational impact, promoting a collaborative culture model.

As hinted, another key factor for the empowerment of e-learning services, is their integration with information from other platforms and services, given that a large amount of data is accessible in, e.g., online communities, blogs, discussion forum, messaging services, and social network sites. In this respect, [8] showcases some effective learning analytics techniques to derive knowledge from unstructured and large blobs of

information. Specifically, the paper focuses on tracking students' data to help them succeed. To this aim, the authors investigate the various types of analytics tools that different universities and institutions adopted, to discuss how faculty can exploit such data to monitor and predict the students' performances, to finally enhance them. Moreover, [9] investigates the same large amount of information freely available over the Web, to explore the opportunities of using such data to get enhancements in all the stages of the learning process, not only in assessments. Following the same current, [10] shows an example of how to fruitfully exploit big data for the prediction of students' performances, to the aim of optimizing their careers.

We point out that, currently, social media can play a vital role with respect to e-learning systems and the effectiveness of information is strongly tied to the way we process these data. In this respect, [11] states that the application of big data with e-learning is a hot topic, which has the potential for creating a huge impact on the whole education system.

We also highlight that processing big data in an effective way is only possible relying on complex cognitive computing solutions implementing suited machine learning techniques. Such techniques allow to cope with data characterized by large volume, different types, high speed, uncertainty and incompleteness, and low value density. Going deep in technical details, algorithms and methods (e.g., representation learning, deep learning, distributed and parallel learning, transfer learning, active learning, and kernel-based learning) is out of the scope of this paper, however the interested readers can find a survey of how machine learning is used for big data processing in [12]. Other references can be found in [13], which showcases a large understanding of past, present and future directions in this domain, made through a mapping of the characteristics of cognitive computing, i.e., observation, interpretation, evaluation and decision, versus the so-called V's of big data, i.e., Volume, Variety, Veracity, Velocity and Value.

2.2 The impact of cognitive computing on e-learning

From the side of cognitive computing, there are only recent works that deal with such advanced techniques used in e-learning applications, for example to serve the instructional design process, helping to find personalized learning assets and improving the definition of individual learning strategies or classifying resources. Among these, [14] focuses on recommendation systems implemented on the basis of cognitive services. The paper envisages the possibility of using the same approach to meet the needs of students and teachers, especially to enable personalized learning strategies and implement recommendation

systems for educational resources, based on information derived from the interactions between students. The authors propose the prototype of a platform and survey the approaches to develop advanced TEL solutions, analyzing the state of the art of using cognitive systems in e-learning, identifying paradigms and pedagogical methodologies, techniques, tools and learning objects with respect to the recommendation of pedagogical activities using cognitive computing. In [15], the authors use cognitive systems for the automated classification of learning videos, with special reference to MOOCs, i.e., exploiting the capabilities of Automatic Speech Recognition (ASR) and Optical Character Recognition (OCR) services to extract text from audio and visual frames so to be able to perform classifications based on taxonomies. Further developments of this work led the authors to find a solution overcoming traditional term-based methods, which analyzes the content of large video collections by means of cognitive services such as: (i) Speech-to-Text tool to get video transcripts, and (ii) the use of Natural Language Processing (NLP) methods to extract semantic concepts and keywords from the above video transcripts [16].

2.3 Empowering e-learning methodologies

Besides, from the methodology perspective, [17] introduces the idea of *big education*, applying the paradigm of big data to the whole education process to predict students' performances, based on individuals' learning attitudes and their after-school activities too. This seems a promising vein, since also [18] theorizes about cognitive analytics driven personalized learning, which can be achieved owing to the advances in cognitive computing for analyzing unstructured data such as, e.g. blogs, discussions, e-mail, and course messages, to gain insights into student learning at an individual level.

New functionalities are strictly connected to new technologies such as, for example, the mix of learning and semantic technologies through the use of ontologies for the description of domains (see, e.g., [19] and references therein) and the availability of sophisticated cloud infrastructures is required to handle properly such a huge quantity of information, as well as the design and development of new learning environments, supporting suited machine learning technologies as reported in [20].

2.4 Privacy and security issues

Finally, we observe that when considering big-data-capable learning applications another paramount item emerges, that is the students' data protection. In fact, personal information in the e-learning frameworks can be very detailed, thus very precise profiling can be obtained and used maliciously for different scopes, such as, e.g.,

remarketing. As well as privacy, security cannot be neglected and should be considered one of the most important factors in the design of TEL platforms [21]. For example, this topic was faced in [22], which clearly presents possible threats of considering big data within e-learning platforms.

3. Emerging trends in TEL

According to the above considerations, we propose a new point of view on TEL platforms, enlightening the main characteristics they should offer. Their design must consider the new wave of cognitive services for their use in advanced solutions and for their ability to cope with the huge amount of data circulating within the learning frameworks and the connected software environments outside the LMS (e.g., discussion forums and blogs, social network sites, indexing services, digital libraries, etc.). In more detail, personal data about learners, their learning tasks, scores in the assignments, etc. should be stored and historicized to the aim of improving the whole learning and teaching process, also including administrative information to monitor how learning processes are conducted and make an assessment to the aim of predicting performances through suited learning analytics. More precisely, we state that the next-generation TEL platforms will offer pervasive cognitive services.

From the side of the learning process, we have to consider two different aspects: (i) learners will acquire new knowledge thanks to the educational strategy and methodology adopted, and (ii) at the same time, machines, systems and platforms will acquire information about learners and their individual learning processes, owing to cognitive services. However, thinking of cognitive services merely as the result of an algorithm is not a good starting point: there is something more because learning does not derive from a software algorithm but also from the complex hardware architecture it relies on. In fact, cognitive services can be effective only within a parallel architecture, whose capabilities must be suited to the learning needs of the algorithm itself. In other words, since the algorithm must be trained, we often need substantial memory and computational resources to be implied in educational processes based on a cognitive approach. The personalization of cognitive services may require additional hardware and software resources and thus, for this reason, when defining a cognitive TEL, we have to go beyond a basic hardware configuration to face to possible personalization issues, which may require a system flexible enough to adapt over time its characteristics to the use we will make of it, in the perspective of a lifelong learning support to our knowledge growth. Such features can be achieved through

the adoption of cloud architectures, Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) solutions. Furthermore, feedback is needed to improve the effectiveness of this approach, and we should be able to observe the system and get feedback to perform real-time tuning operations.

In the light of the above considerations, it is required to build innovative architectures for the platforms that aim to provide innovative didactics, able to change configuration quickly in order to adapt its functionalities to the different needs of both learners and teachers. In particular, we believe that a wise adoption of the best of some services from multiple vendors, in the same platform, could improve the satisfaction of end users as well as solve some structural problems of the laboratories. In fact, often university laboratories need to be scalable for different number of students attend classes or they may need to serve different educational activities (e.g., quizzes, lessons, exercises, application laboratories, exams, etc.). Sometimes they must even be scaled up to more applicative situations, i.e., performing tests for research projects or experiments. Specifically, Section 4 provides a detailed description of the innovative learning environment that the Federico II University in Naples, Italy, setup, based on the above concepts.

Moreover, we observe that e-learning platforms and applications continuously change together with learning itself and learners' attitude and needs. In many cases, educational paths are designed in collaboration with institutions from other countries, i.e., to promote mobility of both workers and researchers, in other cases a strict interaction with industries and the labor market is required. Consequently, performing educational tasks in a "classroom", whether real or virtual, may be limiting the perspective of what today a smart education should be. Hence, in order to stay competitive, formal education models should expand their boundaries also involving the external world and this can only be achieved through the adoption of suited technological solutions for online collaboration, interoperability, data exchange, and the seamless integration with legacy systems. The implementation of such solutions requires suited machinery and infrastructures as well as a paradigm shift able to drive the transformation of classrooms into communities to enhance humans' faculties and empowering the transformation of their skills so that, in such new "places", we can speak of men *and* machines, rather than men *or* machines. Even in this case a specific example is presented and Section 5 depicts the characteristics of the innovative e-learning platform developed at the University of Trento, Italy. platforms.

4. An integrated environment for exponential learning

This section provides a detailed description of the innovative learning environment setup at the Federico II University in Naples, Italy, based on cognitive computing to implement exponential learning. We recall that cognitive computing was born based on human reasoning models and this special capability can be exploited transversally on, potentially, any domain of application, so that it can be regarded as a real revolution, exponentially accelerating processes. In the past decades, we have been accustomed to a swirling growth, according to the Moore's law (the number of transistors in a dense integrated circuit doubles about every two years) or the analogous Metcalfe's law for the number of nodes of a network that clearly explains the enormous dimensions reached by social networking sites such as, e.g., Facebook. However, we were not prepared to cope with the needed exponential growth of learning caused by the growth of knowledge, which is in turn derived by the growth of data processed with novel Artificial Intelligence (AI) techniques. Consequently, the learning environments should be promptly adapted to this reality.

4.1 The hardware and software setup

According to the above considerations, the Federico II University in Naples, Italy, setup a laboratory with suited hardware and software configurations, providing students of the "Software engineering" class with the possibility to use the most advanced methodologies and tools to develop quite complex projects within their assignments and to share their results with mates. Previously, the authors already carried on didactic activities focusing on computer programming in collaborative environments (see, e.g., [23], [24], and references therein) but cognitive computing has accelerated this need, thus enabling more ambitious projects to be implemented both from the point of view of the hardware resources that can be used and the complexity of the software that can be used. The implemented solution is an integrated environment offering high-level tools for cognitive computing design and programming. Specifically, after a preliminary analysis and a series of stress tests, Microsoft Azure and IBM Cloud have been identified as the best-matching solutions for developing cognitive-computing-based projects. Owing to their cloud-based architecture, both platforms can coexist in the same installation, where they can be used simultaneously if needed, allowing to realize the needed PaaS and SaaS infrastructures.

More precisely, students can use the cognitive e-learning platform from both IBM Cloud and Microsoft Azure with relevant laboratory and customized virtual machines. Taking advantage of such a mixed

configuration, enables students to disregard hardware and software issues, focusing on their individual learning tasks. Besides, they can use complex machines simply through a browser, which is a strong point because they may have to use laptops or even personal computers with poor performances. To better clarify, we highlight that the Azure environment offers different profiles for sizing the hardware machine. One can choose one of the following: (i) courses, (ii) laboratories, and (iii) exams. Then, the decision to instantiate a laboratory session or to take part in an exam session is up to the user. The only requirements are the availability of a PC and a network connection. In more detail, the teacher can, in a laboratory consisting solely of the PCs that students bring with them, instantiate an exam session on the fly. Then, the students will have to use their institutional credentials to log into the system, where they can access a personalized dashboard and choose which activity to carry on such as, e.g., laboratory, exams etc. Even if a student is taking an exam, in the same classroom, at the same time, other students can perform different activities. For example, writing and compiling a Java program within the Eclipse IDE as well as entering the IBM Cloud platform, and also use, e.g., the IBM Watson cognitive services.

4.2 Some results

At the end of the activities, the platform allows students to assess the effectiveness and ease-of-use of the environments they used. In particular, they were asked to provide their overall feedback on satisfaction and usability. In order to evaluate the global satisfaction of the platform we considered an ordinal scale with values between 5 and 8. The students were asked to express their opinion after working on the platform for several hours. More precisely, Figure 1 shows that 87% of users gives a medium-high evaluation (scores 7-8), while only 13% believes that the platform has low-sufficient usability (scores 5-6).

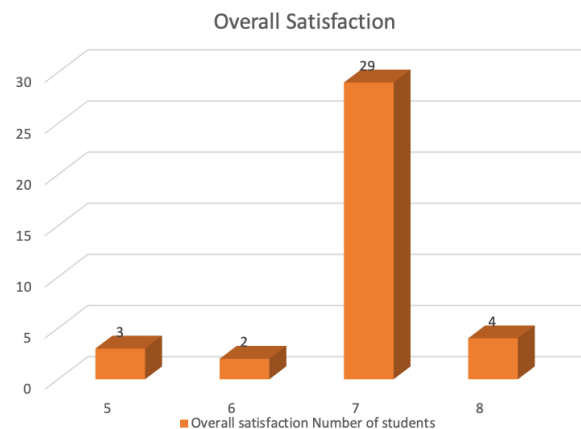


Fig. 1. Results from the questionnaires on users' satisfaction.

Since these integrated environments of multiple technologies may be too complex for young students, we have also asked a feedback on usability whose results are shown in Figure 2. In order to measure the usability of the platform we used an ordinal scale of values between 5 and 9. Students, in a slightly lower number (36 vs. 38), expressed their opinion about the usability of the platform after working on it for the didactic projects they were engaged in. More precisely, Figure 2 shows that the 61% of students gives a medium-high evaluation (scores 7-8-9), while only 39% (scores 5-6) believes that the platform has low-sufficient usability. Besides, students were also asked to express suggestions and possible improvements. From this analysis it emerges that students experienced some difficulties, due to the presence of two different environments but also for the functionalities, rather different from the classical university-laboratory setup they are used to, which offer only basic tools.

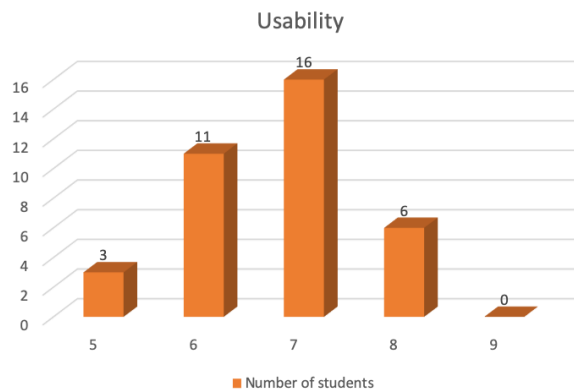


Fig. 2. Results from the questionnaires on usability.

In conclusion, the integrated environment in use was judged satisfactory with respect to the functions it had to perform. Given its apparent complexity, we consider such results quite positive and encouraging for the prosecution of this experiment also in the next academic years, in new classes and with a growing number of students.

5. From classrooms to communities

According to the above considerations, in this section we depict the characteristics of an innovative e-learning platform developed at the University of Trento, Italy. As highlighted in previous sections, classrooms should shift to communities and in such new places both students and teachers will be able to enhance humans' faculties and empowering their skills transformation, also owing to modern technologies and new paradigms for the possible relationships between men and machines. Specifically, we will discuss the Online Community (OLC) project.

5.1 The online community (OLC) project

The project started back in 1998 with the idea of implementing a different approach to educational content management, in contrast to proprietary platforms like WebCT™ and Blackboard™, which were dominant at that time. In this scenario, OLC was created from scratch, after having considered three possible alternative solutions: (i) adopting a commercial platform; (ii) adapting free-open source software to the needs of educational environments, as Moodle or similar where not existing yet (iii) building a brand-new platform. Finally, the third option, despite having the traditional pros and cons of every “make” solution, was chosen for other motivations that now, in the idea of innovating e-learning systems with new cognitive services, revealed to be the right decision. Compared to the adoption of commercial software, the Total Cost of Ownership (TCO) of this kind of solution was probably too high especially in the hypothesis of an extension, not counting the rest of the associated costs for maintenance, management and training, when compared to budget limits. We observed in the following years that this is a very common reason for the adoption of Free and Open Source Software (FOSS) platforms such as, e.g., Moodle by many small-medium educational institutions like high schools and universities. Put simple, being free of charge is the main reason for adopting this solution, without taking care of the side effects on the educational processes that it implies. In fact, a software platform implements processes in a rigid way once they have been coded, so the adoption of this component forces the institution to customize, adapt or even change the way things can be done because “the system does not allow me to do differently”. As a secondary effect, the connection of the LMS with the rest of the organization's information system is mostly impossible [25]. As a consequence, many system administrators, are adapting their needs to the software system that, somehow, is able to solve most of their problems, and they mostly are resistant if not reluctant to develop an internal solution. Money, availability of qualified resources, short time to implement the solution, these all are comprehensible reasons for choosing the easy way of acquiring a pre-cooked solution.

We consider much more important to have the possibility of investing in a platform that can be easily extended with new services according to the needs of those trainers willing to experience the use of computer technologies in their educational processes. We therefore decided to develop a completely new platform, and very soon the platform was transformed from a mere LMS to a more structured platform devoted to support collaboration among members of a virtual community. The idea of “classroom” that is lying behind most of the LMS available today is, in our opinion, very restrictive with

reference to the more complex processes that normally happen inside educational tasks, and can be extended to any other collaboration environment where collaboration among participants to the community are mediated by technologies. In other words, the idea of a customized LMS that could constitute a competitive advantage for one university versus another one is established [26]. This personalized software is able to supply better and personalized services that ease procedures and processes for the different users such as students, professors, or administrative personnel. Finally, a second decisive item was the need of deep integration with the rest of the legacy information system: authentication with single sign-on, integration with exam records and administrative procedures, possibility of bi-directionally exchange news and messages among people living the university day-by-day routine. Substantially, creating the platform from scratch was related with the rejection of “one-size-fits-all” approach to software components of an information system. Many administrators of the information system, especially in the educational sector, are adapting their needs to the software system that, somehow, is able to solve most of their problems, and they mostly are resistant if not reluctant to develop an internal solution. Money, availability of qualified resources, short time to implement the solution, these all are comprehensible reasons for choosing the easy way of acquiring a pre-cooked solution.

This adaptation of educational tasks to software platforms is a typical situation where many institutions are lying today. Because teams not always have the knowledge and resources to modify existing (open source) software educational platforms (i.e., LMSs like Moodle), they normally “adapt” themselves to what the platform supplies out-of-the-box, thus limiting the innovation potential of their ideas, and forcing users to adapt their learning processes to the improper technological tool. The typical example is the use of social media (the one that is more appreciated by learners in that moment) to support educational tasks. Social media such as, e.g., Facebook, Twitter, WhatsApp and Instagram are great tools when applied to the context they have been originally created, mostly exchanging multimedia information among peers. Yet, it is not so easy to integrate them in the educational processes: it is clearly a technical problem of available software Application Programming Interfaces (API), but it is also an instructional design problem introducing issues on how to cope the style of the lecture, how is changing the role of the teacher and what are the expectations of learners about the use of social media. In fact, an educational process is something wider than posting a photo or retweeting others’ comments, even if it can benefit of this situation. Sometimes, educational processes need the support of other tools and services, that Facebook (for example) can provide through

a distorted usage of its services. This normally forces users (mainly educators) to adapt their learning processes to what the platform provides, while it should be exactly the opposite, i.e., the platform should be able to adapt its services to the users’ needs. The same happens with LMSs: most of the educational organizations have no possibilities of intervention, nor adaptation or modification on software platforms that have the size and complexity of Moodle, and so they adapt their educational processes to what the chosen platform provides as standard services.

5.2 Latest innovation

The innovative aspect that we introduced with OLC, and that now constitutes an extra advantage, is therefore reverting this master-slave approach that sees software platform as masters and end-users as slaves. The approach followed by OLC is to construct from scratch those services identified as relevant by educational experts, based on the precise educational needs of the different users: teachers, students or any other role involved. OLC has some architectural aspects there are very important for our argumentation about a next generation of TEL platforms: (i) we own and have created every single line of the source code, so the whole knowledge of the platform architecture and its potential are not scattered among different contributors (like in many open source projects); (ii) the platform is equipped with a micro service architecture, thus allowing an easy extension of the platform itself with new parts but taking advantage of the many services that any LMS should provide both to users and to developers that want to extend it; (iii) some services have been already developed in the past towards the direction of providing “intelligent” services. Owning and knowing the whole source code of the platform means to have a great advantage if you want to extend it with innovative elements, so the idea of using OLC as a basis for cognitive computing has been straightforward. Nevertheless, some crucial evolutionary changes had to be applied to the platform, and these changes are the key success factors for this shift:

a) stimulate interaction: the platform should encourage users to interact, not just to download files with teacher’s slides. Today, the vast majority of LMSs are used solely to download files, while interactive and more participative services are left to the availability in the platform (if any) and/or at the goodwill of the teacher;

b) pervasive and enriched logging: the platform should log actions of users in order to activate cognitive processes: logging is essential for cognitive computing in order to classify users and infer the best service at the best moment. This logging should not be just the web application server logging, but specialized, application-

level logging are needed to capture specific actions inside the single page of the LMS;

c) extensible, service-based architecture: the platform must be extensible through a service-by-design approach, in order to add new services whenever new possibilities can be explored. A micro service architecture is highly recommendable;

d) inference-oriented persistence layer: in order to facilitate inference, reasoning and cognitive computing algorithm, the persistence layer of web platform should be updated to more efficient and flexible data structures [20].

On this basis, a set of profound re-engineering operations has been implemented inside OLC. For example, we extended the platform towards a semantic representation of the knowledge inside the contents of the platform. We also integrated some soft computing, fuzzy-logic-based decision support systems [1], to support decision makers with intelligent tools about educational processes. Moreover, we experimented new storage layers, in order to collect data not only from traditional sources inside the educational environment (i.e., files, forums, galleries, posts, photos, etc.), but also collecting a lot of analytical information about the use of the platform and its services by the users. This immediately opens the problem of the size and appropriateness of traditional relational databases. We performed some experiments in substituting some parts normally stored in relational tables into triples available for a semantic knowledge representation. This meant using triple stores in the beginning as a persistence layer, thus facilitating operations like inference, reasoning, machine learning, etc. The triple (or quadruple) format for persisting (part of) data relevant for decision making and cognitive computing is another step that is not currently available in mainstream LMSs.

To better clarify the above concepts, let us consider as an example the diagram sketched in Figure 3, which depicts the preparation process of proper data sources for big data integration and analysis. The first step is the selection of the data from the persistence of OLC. This persistence is a typical big data source, with structured and unstructured information (file, learning objects, blog posts, forum topics, wikis, etc.) created inside the platform itself. The idea is to separate such data from the rest of the platform, to create the background to be able to apply the cognitive algorithms. So, in a sense, this resembles an Extract Transform and Load (ETL) process, typical in any data warehouse as well as OLAP and data mining solution. The most part of information is coming from the first data source we used in our experimentation, due to its affinity with big data sources, that we call “Actions”. This service collects all data coming from users’ interactions with other OLC objects or services. In practice, it acts like a sensor introduced inside the source

code of the platform in any place the software needs to capture an “action” from the user interface. This is a relevant enrichment of the logs recorded by the web application server, and has been used for many different purposes. Due to volume issues, the system at the moment is blocked on collecting only some types of events, to a certain granularity defined by the system administrator. This choice has not been a design choice, but a performance-related one. In fact, it was clear from the early experimentations that the amount of data could have been compromising the capacity of the DBMS to stand with data acquisition pace and volumes: that is a typical “Velocity” an “Volume” big data problem.

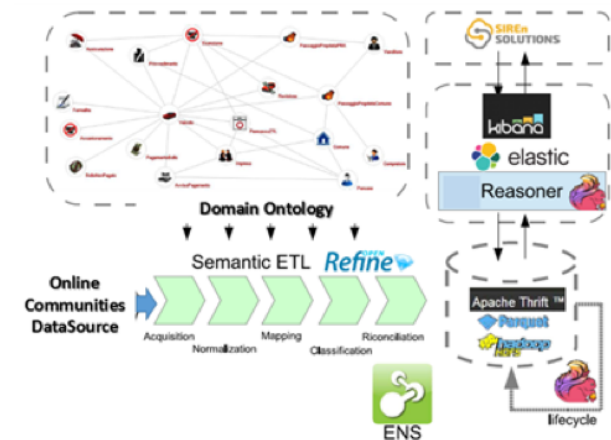


Fig. 3. Overview of the big data preparation process for the analytics tools.

In more detail, there are several elements of data gathering and manipulation pushing our virtual learning environment towards big data, thus increasing the need of a structural change of LMSs architecture, which should adopt new approaches and technologies. Specifically, we mention:

- traditional weblogs, being the application a Web-based software;
- internal logs of usage of the platform, the so-called digital breadcrumbs, that track the learner’s journey throughout the entire learning experience;
- mobile logs, where data about mobile learning actions are collected;
- service logs, users’ actions on the different elements of the platform like documents, forums, blogs, FAQs etc.;
- logs from the SCORM player, normally an external entity respect to the core services of the platform, with the records of the objects’ execution;
- X-API calls, in case the platform is connected or acting as a Learning Record Store (LRS);
- MOOCs, by definition a generator of high volumes of data;

- lifelong learning, an old buzzword of e-learning that is still valid and interesting and, most of all, is another generator of big data, specifically along time;
- serious games that will use materials inside the platform, thus generating a relevant dataset related with users' performances.

The next step in the pipeline is the execution of cognitive tasks which can materialize implicit knowledge to support learning processes. At the moment, this reasoning is limited to basic inference regarding actions performed on certain parts and services of the platform, but the mechanism is ready for larger application scenarios. When the inference process is complete and new knowledge is inferred, a set of administrative routines is executed to load and transform part of the knowledge base to feed applications persistence.

6. Conclusions

As a conclusion, we remark that nowadays we are on a turning point in which cognitive-computing-based approaches are significantly transforming many aspects of our lives. In fact, beside the changes we can notice in professional applications and high-end software and legacy systems, we are already experimenting a set of cognitive computing services in everyday activities, for example, through natural user interfaces and voice assistants, whose presence is becoming pervasive. In practice, more often than we think, we are interacting with machineries that apply sophisticated decision-making processes with very low time constraints and a high level of accuracy. E-learning is one of the fields of application that can mostly benefit from this situation, due to its complexity and to the variety of disciplines that must be adopted concurrently to achieve good learning outcomes. Especially, the use of big data strongly empowers the process of personalization and individualization of the learning processes. Moreover, e-learning is also called to provide suited solutions to the problem of learning to use and exploit such new technologies, which cannot be achieved in environments designed for generic purposes. This raises the problem of developing a new generation of TEL platforms.

The paper introduces the vision of the authors, where a self-made, highly customizable virtual community platform will be integrated with scalable, top-notch cloud platforms and congruent cognitive algorithms applied to the different parts of learning processes, from material selection to educational path suggestions, from peer evaluation to big data discovery for decision makers. The process is still in its infancy, mostly because these three worlds (TEL platforms, cloud services and cognitive computing) are still separated and mostly focus on their

own scope. What we are trying to do is merge the three disciplines into one single research area, with precise objectives and deliverables, thus allowing e-learning to maximize the advantages of the fusion of the three.

The two experiences presented in this paper can be regarded as an embryo for the development of future, unpredictable e-learning solutions. To this aim, the results achieved while using the integrated environment for exponential learning deployed at the Federico II University convinced us to scale the process on a wider and even younger audience, in order to test the real simplicity of such a TEL environment. Another lesson learned is that the surrounding world has become so complex and the changes so rapid that they will never be as slow as in the past. We should not waste our precious time any more in installing software, configuring applications, customizing solutions, tuning databases, maintaining laboratories, etc. Students and teachers definitely have to devote their intellectual power to more creative and less repetitive activities. Moreover, the applications we started using will increasingly have to analyze large amounts of data and therefore be cognitive. Future developments of the OLC also will follow in the direction of providing more advanced cognitive services and making the best from the available information mixing data from different online sources.

References

- [1] Bouquet P., Molinari A. (2016) "A new approach to the use of semantic technologies in e-Learning platforms". *International Journal of Advanced Corporate Learning*, Vol. 9, No. 2, pp. 5–12.
- [2] Anshari M., Alas Y., Sei Guan L. (2016) "Developing online learning resources: Big data, social networks, and cloud computing to support pervasive knowledge". *Education and Information Technologies*, Vol. 21, pp. 1663–1677.
- [3] Coccoli M., Maresca P., Stanganelli L. (2016) "Cognitive computing in education". *Journal of e-Learning and Knowledge Society*, Vol. 12, No. 2, pp. 55–69.
- [4] Coccoli M., Maresca P., Stanganelli L. (2017) "The role of big data and cognitive computing in the learning process". *Journal of Visual Languages & Computing*, Vol. 38, pp. 97–103.
- [5] Medrizzi M., Molinari A. (2013) "A multi-expert fuzzy TOPSIS-based model for the evaluation of e-Learning paths". *Proceedings of the 8th Conference of the European Society for Fuzzy Logic and Technology*, pp. 554–558.
- [6] Banica L., Radulescu M. (2015) "Using big data in the academic environment". *Procedia Economics and Finance*, Vol. 33, pp. 277–286.
- [7] Mahmod M.A., Ali A.M. (2018) "Promotion the e-learning success in universities in Baghdad through enhancing their organizational innovative collaboration environment: A qualitative study". *International Journal on Perceptive and Cognitive Computing*, Vol. 4, No. 1, pp. 12–18.
- [8] Dietz-Uhler B., Hum J.E. (2013) "Using learning analytics to predict (and improve) student success: A faculty perspective". *Journal of Interactive Online Learning*, Vol. 12, No. 1, pp. 17–26.
- [9] Kolekar S.V., Pai R.M., Pai M. M. M. (2017) "Prediction of learner's profile based on learning styles in adaptive e-learning

- system". *International Journal of Emerging Technologies in Learning*, Vol. 12, No. 6, pp. 31–51.
- [10] Yu T., Jo I.H. (2014) "Educational technology approach toward learning analytics: relationship between student online behaviour and learning performance in higher education". *Proceedings of the 4th International Conference on Learning Analytics and Knowledge*, pp. 269–270.
- [11] Sheshasaayee A., Malathi S. (2017) "Impact and consequences of big data in e-learning". *Proceedings of the International Conference on Innovative Mechanisms for Industry Applications*, pp. 726–729.
- [12] Qiu J., Wu Q., Ding G., Xu Y., Feng S. (2016), "A survey of machine learning for big data processing". *URASIP Journal on Advances in Signal Processing*, Vol. 67.
- [13] Gupta S., Kumar Kar A., Baabdullah A., Al-Khowaiter W.A.A. (2018) "Big data with cognitive computing: A review for the future". *International Journal of Information Management*, Vol. 42, pp. 78–89.
- [14] Leitão G.d. S., Valentin E. B., Oliveira E. H. T.d., Barreto R.d. S. (2018) "Survey on pedagogical resources recommendation using cognitive computing systems". *Proceedings of the IEEE Frontiers in Education Conference*, pp. 1–7.
- [15] Dessì D., Fenu G., Marras M., Reforgiato Recupero D. (2017) "Leveraging cognitive computing for multi-class classification of e-learning videos". In Blomqvist E., Hose K., Paulheim H., Ławrynowicz A., Ciravegna F., Hartig O. (eds), *The Semantic Web: ESWC 2017, Lecture Notes in Computer Science*, Vol. 10577, Springer, Cham.
- [16] Dessì D., Fenu G., Marras M., Reforgiato Recupero D. (2019) "Bridging learning analytics and cognitive computing for big data classification in micro-learning video collections". *Computers in Human Behavior*, Vol. 92, pp. 468–477.
- [17] Cen L., Ruta D., Ng J. (2015) "Big education: opportunities for big data analytics". *Proceedings of the IEEE International Conference on Digital Signal Processing*, Singapore, pp. 502–506.
- [18] Gudivad V.N. (2017) "Cognitive analytics driven personalized learning". *Educational Technology*, Vol. 57, No. 1, pp. 23–31.
- [19] Coccoli M., Torre I. (2014) "Interacting with annotated objects in a Semantic Web of Things application". *Journal of Visual Languages & Computing*, Vol. 25, No. 6, pp. 1012–1020.
- [20] Bouquet P., Molinari A. (2012) "Semantic technologies and e-learning: Towards an entity-centric approach for learning management systems". *Journal of E-Learning and Knowledge Society*, Vol. 8, No. 2, pp. 65–84.
- [21] Caviglione L., Coccoli M. (2018) "Smart e-learning systems with big data". *International Journal of Electronics and Telecommunications*, Vol. 64, No. 4, pp. 445–450.
- [22] Habegger B., Hasan O., Brunie L., Bennani N., Kosch H., Damiani E. (2014) "Personalization vs. privacy in big data analysis". *International Journal of Big Data*, pp. 25–35.
- [23] Coccoli M., Maresca P., Stanganelli L., Guercio A. (2015) "An experience of collaboration using a PaaS for the smarter university model". *Journal of Visual Languages & Computing*, Vol. 31, pp. 275–282.
- [24] Maresca P., Guercio A., Stanganelli L. (2012) "Building wider team cooperation projects from lessons learned in open communities of practice". *Proceedings of the 18th International Conference on Distributed Multimedia Systems*, pp. 144–149.
- [25] Colazzo L., Molinari A., Villa N. (2009) "Social networks, virtual communities and learning management systems: Towards an integrated environment". *Proceedings of the 8th IASTED International Conference on Web-Based Education*, pp. 209–215.
- [26] Kimball L., (2002) "Managing distance learning: New challenges for faculty". In *The Digital University - Building a Learning Community*, Springer London, pp. 27–40.

Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jvlc

Comprehension of Software Architecture Evolution supported by Visual Solutions: A Systematic Mapping and a Proposed Taxonomy

Joao Werther^a, Glauco de Figueiredo Carneiro^b and Rita Suzana Pitangueira Maciel^a

^aFederal University of Bahia, Salvador, BRA

^bUniversidade Salvador (UNIFACS), Salvador, BRA

ARTICLE INFO

Article History:

Submitted 8.1.2019

Revised 8.20.2019

Second Revision 8.20.2019

Accepted 8.20.2019

Keywords:

software architecture
software visualization
software architecture evolution
software architecture comprehension

ABSTRACT

Context: Software visualization has the potential to support specialized stakeholders to understand the software architecture (SA) evolution. To the best of our knowledge, there is no guideline to support the use of visual solutions towards SA evolution comprehension. **Goal:** Analyze the use of visual solutions for the purpose of comprehension with respect to software architecture evolution from the point of view of software architects and developers in the context of both academia and industry. **Method:** We conducted a Systematic Mapping Study to achieve the stated goal. **Results:** The study identified 211 papers published from January 2000 to May 2019 as a result of the search strings execution. We selected 21 primary studies and identified a gap in terms of a taxonomy to assist specialists in the development or classification of solutions to support the comprehension of software architecture evolution using visual resources. **Conclusion:** We observed that despite the relevance of the use of visual solutions to support the comprehension of software architecture evolution, only 21 studies have reported these initiatives, suggesting that there is still room for the use of different visual metaphors to represent its components, relationships and evolution throughout the releases.

© 2019 KSI Research


1. Introduction


Software evolution reflects changes undergone by the software during its lifespan [1] [2]. The study of software evolution is essential to better support changes in software requirements over time, keeping its integrity at a lowest possible cost [3] [4] [5]. Software Architecture (SA) is the design model used to build and evolve a software system [6]. Throughout the analysis of the software architecture, it is possible to understand the dimensions along which a system is expected to evolve [1]. The importance of SA in software evolution process is that a software built without an adaptable architecture normally will degenerate sooner than others with a change-ready architecture [1]. The evolution of a SA can be the result of changes in the current SA to accommodate business demands, new technologies and/or

platform or other reason that impacts the SA [7]. The comprehension of SA is essential for the development and evolution of software systems [8][9] and can be supported by software visualization resources to understand key SA characteristics regarding architectural models and design decisions [8] [10].

Software Visualization (SV) has been used to support the SA comprehension in the context of software systems. This support usually occurs through the use of different visual metaphors to represent its components, relationships and evolution throughout the releases [9]. The use of visual solutions to represent SA and its architecture design decision can improve significantly their understanding [9]. SA visualization may concern with the evolution of its components throughout the releases, not only its static visualization [11].

To the best of our knowledge, there is no guideline to support the use of visual solutions towards SA evolution comprehension. For this reason, we conducted this Systematic Mapping Study (SMS) to identify evidence in the literature on this issue provided by papers published in peer-reviewed conferences and journals from January 2000 to May 2019.

 jwertherf@gmail.com (J. Werther); glauco.carneiro@unifacs.br (G.d.F. Carneiro); ritasuzana@dcc.ufba.br (R.S.P. Maciel)

 www.unifacs.br (G.d.F. Carneiro); pgcomp.dcc.ufba.br (R.S.P. Maciel)

ORCID(s): 0000-0001-6241-1612 (G.d.F. Carneiro);
0000-0003-3159-6065 (R.S.P. Maciel)

DOI reference number: 10.18293/JVLC2019N1-008

From the 211 studies retrieved by the search string applied in specific electronic databases, we selected 21 studies to gather evidence to answer the stated research questions.

We aim at identifying strengths, weaknesses and research gaps related to the use of visual solutions to support the comprehension of software architecture evolution. Additionally, the analysis of the selected papers allowed us to identify the opportunity to propose a taxonomy to characterize and evaluate visual solutions to support the comprehension of SA evolution, representing their main characteristics, properties and features. According to Price, Baccker and Small (1993) [12], a well-founded taxonomy provides a common terminology and a set of related concepts that facilitate the communication and classification of information in a specific area, enabling the identification and cataloging of new discoveries and ideas in this area. The selected studies were classified using the *category* dimension of the proposed taxonomy as follows: 14% were categorized as *Description*, 38% as *Technique*, 67% as *Tool* and 19% as *Environment*. Besides the *category* dimension, the proposed taxonomy contains other five additional dimensions: *stage*, *visualization form*, *static representation*, *dynamic representation* and *architectural tasks*.

This paper is an extension of an earlier conference paper [13]. Our original work related a systematic mapping conducted for analyze the use of visual solutions for the purpose of comprehension with respect to software architecture evolution from the point of view of software architects and developers in the context of both academia and industry. In this extension work, a new version of this systematic mapping, we adjusted the PICO criteria to build a new version of the search string maintaining the original goal and research questions. We aimed at increasing the number of selected papers and therefore improve the findings discussed in this study. We also included a new background section to present a contextualization of issues related to software architecture, software evolution, architecture evolution, software visualization and software architecture visualization considered relevant in this systematic mapping. We also narrowed the interval of publication of searched papers by changing the upper bound of the interval from December 2018 to May 2019. However, we did not identified impact of this change in the number of retrieved papers.

The remainder of this paper is organized as follows. Section 2 shows a brief contextualization on software architecture, software architecture evolution and their respective relationships with software visualization. Section 3 discusses related works and Section 4 presents the design we adopted to conduct this SMS. The Section 5 reports the results and findings of this study, and proposes a new taxonomy for visual solutions to software architecture evolution. The Section 6 presents the answers to the stated research questions. Finally, we conclude and mention future work in Section 7.

2. Background

SA refers to a set of components of a software system, their connections and their principles and guidelines to manage the development and evolution during software life cycle [14]. SA describes the system's structure, interaction of its components and their core properties, playing an important role as an interface between requirements and source code [15]. SA is a possible mean to provide evidence if in fact the software is in compliance with its non-functional requirements (e.g performance, reliability, scalability, etc.) [15]. SA is also used to describe high-level structures and behavior of a software system [16], which contributes to support the software evolution [17]. In addition, SA provides a better understanding of the software to its stakeholders [15].

Software evolution refers to the dynamic behavior of software systems as they are submitted to changes over time [18] [19]. The analysis of software evolution is essential to both understand past and to plan future changes in the software, keeping its integrity (mainly the architectural one), at a lowest possible cost [20]. The importance of SA in software evolution process is that a software built without an adaptable architecture is prone to have shorter lifespan than others that have a change-ready architecture, impairing its evolution over time [21]. The analysis of SA evolution significantly improves the perception of software evolution. SA allows the planning and restructuring of the software system in a high level of abstraction, being a valuable reference for the discussion with stakeholders regarding the quality and business trade-off [7]. The comprehension of SA is essential for the development and evolution of software systems. Due to the amount of information, this task can be more effective when supported by resources that help to soften the required cognitive effort to perform it [11].

Shahin, Liang and Khayyambashi [9] argued that the use of SV to represent SA and also its evolution can improve the comprehension of both. In fact, SV provides solutions to support SA comprehension and its corresponding evolution [8]. The SA visualization is an important area in SV [11] and visually represents components and structures from a given SA associated with its architecture design decisions [6]. It may involve not only the visualization of software structures and their relationships, but also the evolution of these structures in the software lifespan [11]. Gallagher, Hatch and Munro (2008) [22] stated that SA visualization can improve the comprehension of software systems for all their stakeholders in all their aspects, along their evolution. Besides being fundamental to discuss and understand the SA in accordance with the variety of project stakeholders, SA visualization is also critical for decisions related to SA [23] and can visually represent some architectural design decisions [6]. In the framework proposed by Gallagher, Hatch and Munro (2008) [22] to evaluate SA visualization tools, one of the features of the key area Task Support (TS) is the *TS Show evolution*. The question of this feature using the GQM approach [24] was "*Does visualization show the evolution of software architecture?*". This framework considers that a SA visualization tool should indeed provide facility to show

evolution, whether in basic or advanced way [22].

3. Related Works

Software visualization has been used in different areas of software engineering such as software architecture, software evolution and software design [8][25]. In the following paragraphs we present results provided by a selection of secondary studies that discussed the use of software visualization to support the execution of activities targeting software architecture. This is not an exhaustive list, it is rather an illustrative set of relevant papers that motivated the conduction of this systematic mapping.

Shahin, Liang and Babar [8] conducted a systematic review to characterize the use of Visualization Techniques (VT) to represent SA in different application domains. Results classified the VTs into four types, according to its popularity: graph-based, notation-based, matrix-based and metaphor-based. From this set, the graph-based stood out for its popularity in industry. The same authors argued that VTs have been used to support SA activities for several purposes: (i) the understanding of architecture evolution; (ii) the understanding of static characteristics of architecture; and (iii) search, navigation, and exploration of architecture design [8]. Additionally, the systematic review reported that VTs have been applied to support SA related activities in a large range of domains. From those domains, *software graphics* and *distributed system* have received special attention from the industry. Finally, the authors argued that SV is one of the interesting ways to support the understanding of the rationale behind design decisions that affect software architecture [8]. It should be mentioned that Shahin, Liang and Babar [8] focused on VT to represent SA. They did not discuss the use of SV to support activities related to the comprehension of SA evolution.

Telea, Voinea and Sassenburg (2010) [25] performed a survey to investigate the use of visual tools for the comprehension of SA from the perspective of stakeholders. They analyzed the results using software architecture visualizations tools (AVTs) aiming to guide industrial practitioners in the adoption of tools and techniques according requirements and capabilities of each type. The authors considered three types of stakeholders: technical users (developers), project managers/lead architects, and consultants. They concluded that AVTs were effective to support technical users and less adequate for consultants, according to expectancy of each stakeholder [25]. Although Telea, Voinea and Sassenburg (2010) [25] focused on the use of visual solutions for SA comprehension, they did not focus on SA evolution solutions.

Breivold, Crnkovic and Larson (2012) [1] conducted a systematic review focusing on software architecture evolution. The goal of the review was to provide an overview at the architectural level of existing approaches in the analysis of software evolution, and also examine possible impacts of this theme on both research and industry. The authors identified five main categories related to this theme: (i) tech-

Table 1

The Goal of this SMS according to the GQM Approach

<i>Analyze</i>	the use of visual solutions
<i>for the purpose of</i>	comprehension
<i>with respect to</i>	software architecture evolution
<i>from the point of view of</i>	software architects and developers
<i>in the context of</i>	both academia and industry

niques supporting quality consideration during software architecture design, (ii) architectural quality evaluation, (iii) economic evaluation, (iv) architectural knowledge management, and (v) modeling techniques [1]. The conclusion of this study emphasized the need of development and improvement of methods, process and/or tools to design architecture in large systems, due to the amount and complexity of artifacts produced and used during their respective lifecycle. This study also presented conclusions for researchers and practitioners, including considering the possibility to elaborate new ideas beyond Lehman's laws (about software evolution). Additionally, this paper also reported the existence of only few works targeting the theme, indicating the need of more research effort in this area [1]. Despite Breivold, Crnkovic and Larson (2012) [1] studied the evolution of SA, they did not emphasize how visual solutions can be used to support the comprehension of SA evolution.

We could identify the relevant contribution of the aforementioned studies to the SA area, including SA evolution. However, they did not focus on visual solutions to support the comprehension of the SA evolution.

4. Research Design

We conducted a SMS to find evidence for the use of visual solutions to support the comprehension of SA evolution during the software lifespan. A SMS is a form of a systematic literature review (SLR) with more general research questions, aiming to provide an overview of the given research [26]. We decided to conduct a SMS due to the potential that this methodology has to reduce the analysis bias, through the establishment of selection procedures [27].

4.1. Planning

The protocol we adopted to conduct this secondary study was comprised of objectives, criteria for considering papers, research questions, selected electronic databases, search strings, selection procedures, exclusion, inclusion and quality criteria to select the studies from which we aim to answer the stated research questions [27]. The protocol of this SMS and related artifacts are available in a public Github repository¹. The goal of this study is presented in Table 1 according to the GQM approach [28].

¹<https://github.com/jvlc2019saevolution/submission>

Table 2
PICO Criteria for Search Strings

(P)opulation	studies in software architecture
(I)ntervention	visual solutions to support the comprehension of software architecture evolution
(C)omparison	not applicable
(O)utcomes	solutions (i.e., tools, techniques, environment, approaches, models or methodology) with focus on visual resources to support software architecture evolution; visual solutions to software architecture evolution; use of visual resources to comprehension of software architecture evolution

The Research Question (RQ) is “*How have researchers and practitioners from academia and industry used software visual solutions to support the comprehension of software architecture evolution based on papers published in the peer-reviewed literature?*”. This research question is in line with the goal of this review, and has been derived into four specific research questions, as follows: Specific Research Question 1 (SRQ1): *What are the main visual solutions to support the software architecture evolution comprehension?* Specific Research Question 2 (SRQ2): *What are the purposes of each visual solution to support the software architecture evolution comprehension?* Specific Research Question 3 (SRQ3): *How the solutions designed to visually support comprehension of software architecture evolution can be classified?* Specific Research Question 4 (SRQ4): *Which visual forms are used to support comprehension of software architecture evolution?*

The motivation behind RQ is justified by the acknowledgment that the comprehension of the software architecture evolution is required to tackle issues or improvements related to the software architecture and its evolution throughout releases [23] [6] [9] [8] [7] [3]. The specific research questions have the goal to gather evidence to support the answer of the stated RQ.

We considered the PICO criteria to define the search strings, as shown in Table 2. The search strings are based on this criteria for the selective process of papers for this review.

The formation of the search string applied in the electronic databases is shown in Tables 3 and 4. The Table 3 refers to major terms for the research objectives, built using the PICO criteria. We also used of alternative terms and synonyms of these major terms. For example, the term *visualization* can be associated with terms such as *visual*, *visualizing* and *visualize*. These alternative terms, as shown in Table 4, are also included in the search string. We built the final search string by joining the major terms with the boolean “AND” and joining the alternative terms to the main terms with the boolean “OR”. The focus of the formed search strings is to focus on papers targeting the research questions of this systematic mapping.

Table 5 presents the electronic databases from which we retrieved the papers along with the respective search strings used to retrieve the papers. Table 6 presents the criteria for exclusion and inclusion of papers in this review. The OR connective adopted in the exclusion criteria, means that the

Table 3
Major terms for the research objectives

Criteria	Major Terms
(P)opulation	AND “software architecture”
(I)ntervention	AND “comprehension” AND “evolution”
(C)omparison	Not Applicable
(O)utcomes	AND “visual” AND “solution”

Table 4
Alternative terms from majors terms

Major Term	Alternative Terms
“evolution”	(“evolution” OR “evolve” OR “evolving”)
“comprehension”	(“comprehension” OR “understanding” OR “understand” OR “support” OR “analysis” OR “evaluation” OR “examination” OR “explore” OR “exploring”)
“solution”	(“tool” OR “environment” OR “technique” OR “approach” OR “model” OR “methodology” OR “solution”)
“visual”	(“visualization” OR “visualizing” OR “visualize” OR “visual”)

Table 5
Electronic Databases Selected for this SMS

Database and URL	Search Strings
Scopus www.scopus.com	(“software architecture” AND (“evolution” OR “evolve” OR “evolving”) AND (“comprehension” OR “understanding” OR “understand” OR “support” OR “analysis” OR “evaluation” OR “examination” OR “explore” OR “exploring”) AND (“tool” OR “environment” OR “technique” OR “approach” OR “model” OR “methodology” OR “solution”) AND (“visualization” OR “visualizing” OR “visualize” OR “visual”))
ACM Digital Library portal.acm.org	(+“software architecture” + (“evolution” OR “evolve” OR “evolving”) + (“comprehension” OR “understanding” OR “understand” OR “support” OR “analysis” OR “evaluation” OR “examination” OR “explore” OR “exploring”) + (“tool” OR “environment” OR “technique” OR “approach” OR “model” OR “methodology” OR “solution”) + (“visualization” OR “visualizing” OR “visualize” OR “visual”))
Engineering Village (Ei Compendex) www.engineeringvillage.com	(“software architecture” AND (“evolution” OR “evolve” OR “evolving”) AND (“comprehension” OR “understanding” OR “understand” OR “support” OR “analysis” OR “evaluation” OR “examination” OR “explore” OR “exploring”) AND (“tool” OR “environment” OR “technique” OR “approach” OR “model” OR “methodology” OR “solution”) AND (“visualization” OR “visualizing” OR “visualize” OR “visual”))
IEEE Xplore ieeexplore.ieee.org	(“software architecture” AND (“evolution” OR “evolve” OR “evolving”) AND (“comprehension” OR “understanding” OR “understand” OR “support” OR “analysis” OR “evaluation” OR “examination” OR “explore” OR “exploring”) AND (“tool” OR “environment” OR “technique” OR “approach” OR “model” OR “methodology” OR “solution”) AND (“visualization” OR “visualizing” OR “visualize” OR “visual”))

exclusion criteria are independent, i.e., meeting only one criterion is enough to exclude the paper. On the other hand, the AND connective in the inclusion criteria, means that all inclusion criteria must met to select the paper under analysis. Table 6 also presents the quality criteria used for this review represented as questions that were adopted and adjusted from Dyba and Dingsoyr [29]. A critical examination following the quality criteria established in this table was performed in all remaining papers that passed the exclusion and inclusion criteria. All these criteria must met (i.e., the answer must be YES for each one) to permanently select the paper, otherwise the paper must be excluded. The exclusion,

Table 6
Exclusion, Inclusion and Quality Criteria

Type	Id	Description	Connective or Answer
Exclusion	E1	Published earlier than 2000	OR
Exclusion	E2	The paper was not published in a peer-reviewed journal or conference	OR
Exclusion	E3	The paper does not present a primary study	OR
Exclusion	E4	The paper is not written in English	OR
Exclusion	E5	The paper has less than 3 pages	OR
Inclusion	I1	The paper must present an approach in the usage of visual solution to support the comprehension of software architecture evolution	AND
Quality	Q1	Are the aims of the study clearly specified?	YES/NO
Quality	Q2	Is the context of the study clearly stated?	YES/NO
Quality	Q3	Does the research design support the aims of the study?	YES/NO
Quality	Q4	Has the study an adequate description of the visual solution?	YES/NO
Quality	Q5	Is there a clear statement of findings by applying the visual solution to support the comprehension of software architecture evolution?	YES/NO

Table 7
Steps of the Selection Process

Step	Description
1	Apply the search strings to obtain a list of candidate papers in specific electronic databases.
2	Remove replicated papers from the list.
3	Apply the exclusion criteria in the listed papers.
4	Apply the inclusion criteria after reading abstracts, introduction and conclusion in papers not excluded in step 3.
5	Apply quality criteria in selected papers in step 4.

Table 8
Classification Options for Each Retrieved Paper

Classification	Description
Excluded	Papers met the exclusion criteria.
Not Selected	Papers not excluded due to the exclusion criteria, but did not meet the inclusion or quality criteria.
Selected	Papers did not meet the exclusion criteria and met both the inclusion and quality criteria.

inclusion and quality criteria were used in the selection process as presented in Table 7. According to Table 8, at the end of the selection process, all the retrieved papers were classified in one of the three options: *Excluded*, *Not Selected* and *Selected*.

4.2. Execution

The quantitative evolution of papers throughout the execution of this SMS is summarized in Figure 1. The figure uses the PRISMA flow diagram [30] and shows the performed steps and the respective number of documents for each phase of the SMS, following the outline described in Subsection 4.1.

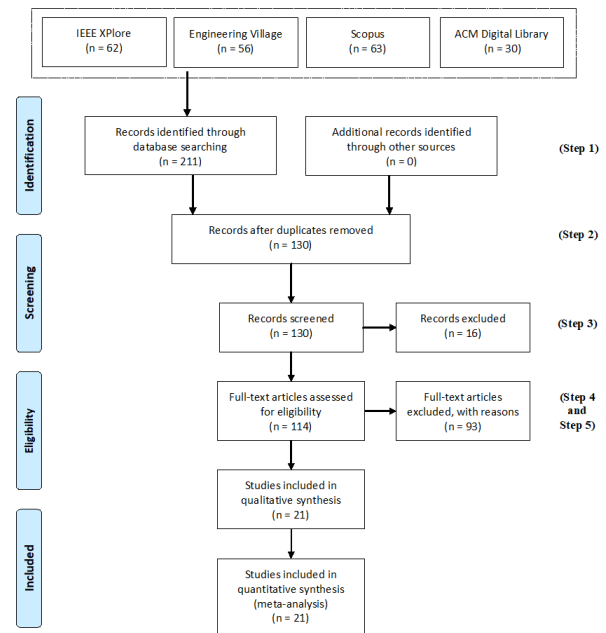


Figure 1: Procedures and its results in the papers selection process.

Table 9
Effectiveness of the Search Strings

Database	Selected Papers	Excluded Papers	Not Selected Papers	Replicated Papers	Total Search Results	Search Effectiveness
ACM Digital Library	3	3	23	1	30	10.0%
Engineering Village	2	3	16	35	56	3.6%
IEEE Xplore	12	5	38	7	62	19.4%
Scopus	4	5	16	38	63	6.3%
TOTAL	21	16	93	81	211	10.0%

Table 9 presents the effectiveness of the the search strings showed in Table 5 considering the 211 retrieved papers. The electronic database that more contributed with selected studies was the *IEEE Xplore* with five papers, corresponding to a search effectiveness of 18.5%. The twelve selected papers represented 13.0% of all 211 retrieved papers.

The Figure 2 presents a overview of contribution of each exclusion criterion in total of excluded papers. The exclusion criterion that had more contribution was the E1 criterion that says “*Published date less than 2000*”, accounting for 50% of excluded papers.

The Figure 3 presents graphics that provide a overview of sources (electronic databases) distribution by papers status. The papers status is according to Table 8. The Figure 3a shows the selected papers distribution by source. The “*IEEE Xplore*” had the major contribution for selected papers with 57% of occurrences. The “*Scopus*” was the second with 19% of selected studies. The Figure 3b shows the not-selected papers distribution by source. The “*IEEE Xplore*” leaded with 41% of not-selected papers and “*ACM Digital Library*” came in second with 25%. The Figure 3c presents the excluded papers distribution by source. The “*Scopus*”

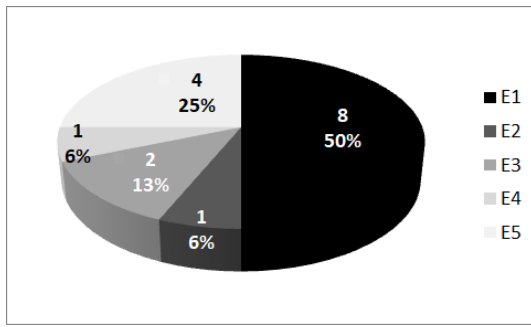


Figure 2: Contribution of exclusion criteria in total of excluded papers.

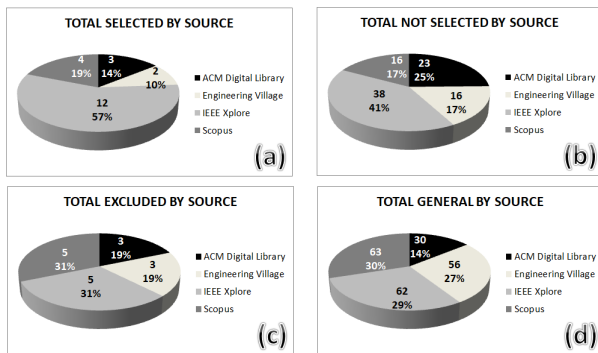


Figure 3: Overview of paper totalization by status and by electronic databases.

and “IEEE Xplore” databases had the majors contributions with 31% of excluded papers each one. The Figure 3d presents the distribution of all found papers by source. The “Scopus” had the major contribution with 30%, close to “IEEE Xplore” with 29% of all papers found in its databases searches.

5. Results

The Table 10 shows the list of 21 selected papers of this systematic mapping. All papers are labeled as “S” followed by the paper reference number. There is 18 selected papers that were published in conferences. The other 3 articles (S02, S03 and S16) were published in journals. They are discriminated in the “Venue” column.

The paper S01 [4] describes a solution aimed at enhancing the comprehension of the software architectural evolution based on visual resources. This solution proposes the use of efficient navigation and visualization of the history of software architectural changes throughout releases, integrating the use of evolution metrics with software visualization techniques. This integration has the goal to support both tracking and analysis of architectural changes from past releases. The authors developed the so called *Origin Analysis* method to analyze software structural change. This method supports the identification of possible origin of function or file that appears to be new in a later release of the software system, if it already existed in the system elsewhere [4]. This method highlights the use of two techniques in its implementation: *Bertillonage Analysis* and *Dependency Analysis*. The

paper also performs a study of evolution of a real tool to demonstrate the use of *BEAGLE*, a prototype implementation of this solution that works as an integrated environment for studying software architecture evolution, as a validation form of the *Origin Analysis* [4]. This paper does not discuss explicitly its limitations, even though cites some of them.

The paper S02 [31] proposes the usage of architecture stability or resilience concepts to evaluate a SA, using *Retrospective Analysis* to achieve this goal. *Retrospective Analysis* is a technique that verify the amount of changes applied in successive releases of a software system and analyze how smoothly the evolution took place. It works with a set of software metrics based in size, growth, changes and coupling, using visual tools to graphically observe the evolution of these metrics. The authors [31] affirms that the *Retrospective Analysis* can have many uses, not only to verify the stability, but also to calibrate the predictive evaluation results as well as to predict trends in evolution of software. The paper describes a case study of twenty releases of a telecommunication software system containing a few million lines of code to show how *Retrospective Analysis* may be performed. However, the paper does not provide any procedures to perform the *Retrospective Analysis*.

The paper S03 [32] introduces a graphical and formal model to represent architecture styles and their reconfigurations in software evolution. The model specifies a SA using graphs and graph grammars to represent components (also called edges) and connections (also referred as nodes). Two techniques are formally presented. The first uses *Synchronized Hyperedge Replacement Systems*, dynamically allowing changes of components and connections according to their synchronization requirements specified in the nodes. The second technique specifies complex reconfigurations as transformations over derivations of graph grammars using *lambda-calculus*. However, the techniques are only described in a formal and summarized way and the paper does not mention any implementation or case study of them [32].

The paper S04 [33] presents the tool-set and methodology *Complex Systems Analysis Based Architecture (ABACUS)* as a visual solution to model complex systems, comprehend their architecture and analyze their characteristics and its potential changes. The paper reports that *ABACUS* allows to collect and merge all enterprise architectural information of a system into a unified repository and also evaluate system properties like performance, openness, and evolvability. The paper also highlights that *ABACUS* provides a hierarchical 3D visualization to allow to look across the enterprise architecture. The authors [33] emphasize that the use of *ABACUS* is not only use its tool-set, but also follow its methodology, as illustrated in Figure 4. They conclude affirming that the usage of *ABACUS* allows the architects conduct the architecture design and evolution based on quantifiable non-functional requirements. However, the paper does not presents any case study of *ABACUS* neither presents any evidence of its practical usage.

The paper S05 [34] proposes a graphical technical description of the architectural instance of a software system,

Table 10
List of Selected Papers

Ref. Label	Title	Venue	Year
S01	An integrated approach for studying architectural evolution [4]	10th International Workshop on Program Comprehension	2002
S02	On architectural stability and evolution [31]	Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 2361, pp. 13-23 (JOURNAL)	2002
S03	Two graph-based techniques for software architecture reconfiguration [32]	Electronic Notes in Theoretical Computer Science, Vol. 51, pp. 177 - 190 (JOURNAL)	2002
S04	The ABACUS architectural approach to computer-based system and enterprise evolution [33]	12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS)	2005
S05	An Approach based on Bigraphical Reactive Systems to Check Architectural Instance Conforming to its Style [34]	First Joint IEEE/IFIP Symp. Theoretical Aspects of Software Engineering (TASE)	2007
S06	Exploring Inter-Module Relationships in Evolving Software Systems [5]	11th European Conference on Software Maintenance and Reengineering (CSMR)	2007
S07	Technology Infusion of SAVE into the Ground Software Development Process for NASA Missions at JHU/APL [35]	IEEE Aerospace Conference	2007
S08	The SAVE Tool and Process Applied to Ground Software Development at JHU/APL: An Experience Report on Technology Infusion [36]	31st IEEE Software Engineering Workshop (SEW)	2007
S09	Visualizing Software Architecture Evolution Using Change-Sets [37]	14th Working Conference on Reverse Engineering (WCRE)	2007
S10	YARN: Animating Software Evolution [38]	4th IEEE International Workshop on Visualizing Software for Understanding and Analysis	2007
S11	Development of a Methodology, Software-Suite and Service for Supporting Software Architecture Reconstruction [39]	14th European Conf. Software Maintenance and Reengineering	2010
S12	Evolve: tool support for architecture evolution [40]	33rd Int. Conf. Software Engineering (ICSE)	2011
S13	Model-Based Software Architecture Evolution and Evaluation [41]	19th Asia-Pacific Software Engineering Conference	2012
S14	eCITY: A Tool to Track Software Structural Changes Using an Evolving City [42]	IEEE International Conference on Software Maintenance	2013
S15	Run-time monitoring and real-time visualization of software architecture [43]	Asia-Pacific Software Engineering Conference, APSEC	2013
S16	eCITY: Evolutionary software architecture visualization - An evaluation [44]	Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 8345 LNCS, pp. 201-224 (JOURNAL)	2014
S17	eCITY+: A Tool to Analyze Software Architectural Relations Through Interactive Visual Support [45]	European Conference on Software Architecture Workshops	2014
S18	The ARAMIS Workbench for Monitoring, Analysis and Visualization of Architectures Based on Run-time Interactions [46]	European Conference on Software Architecture Workshops	2015
S19	Towards the understanding and evolution of monolithic applications as microservices [47]	42nd Latin American Computing Conference, CLEI	2016
S20	Supporting software architecture evolution by functional decomposition [48]	5th International Conference on Model-Driven Engineering and Software Development - MODELSWARD	2017
S21	EVA: A Tool for Visualizing Software Architectural Evolution [3]	40th International Conference on Software Engineering (ICSE)	2018

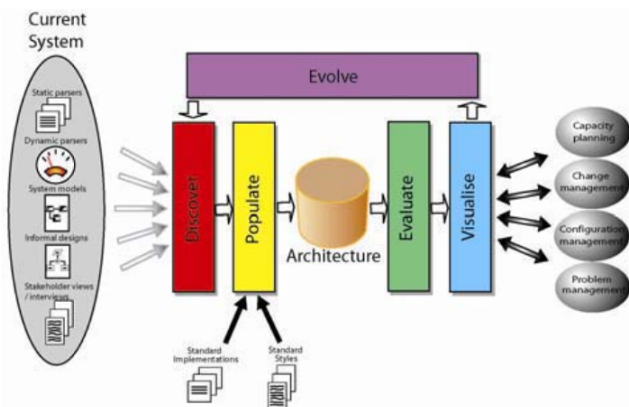


Figure 4: The ABACUS methodology [33]

and verify the compliance to its corresponding style. This solution is based on the *Bigraphical Reactive Systems* (BRS) to perform the verification with formal methods, and uses

an extended version of a Bigraph to describe the instance. Besides supplying a visual method to specify architectural instances and styles, the solution proposed can enhance the ability to design evolving systems. Additionally, the paper shows two study cases in order to prove the effectiveness of this solution [34].

The paper S06 [5] proposes an approach based on the visual representation of inter-module dependencies and relationships between SA components and modules throughout multiple versions of the software system. The *Semantic Dependency Matrix* is a visualization technique that shows dependencies between two modules with similar behavior classes. The *Edge Evolution Film Strip* is a visualization technique that presents the evolution of an inter-module relations in a software system along its multiples versions. These techniques were applied in two large open source software systems, in reverse engineering context, to exemplify them. The paper also purposes a pattern language for inter-module relationships. The studied examples are provided from an exploration prototype named *SoftwareNaut* [5].

The papers **S07** [35] and **S08** [36] describe the NASA JHU/APL's experiences in using the *SAVE* (*Software Architecture Visualization and Evaluation*) tool and process. The *SAVE* tool addresses the understanding, maintenance and evolving issues, allowing software architects to navigate, visualize, analyze, compare, evaluate, and improve their software systems, all in only one environment. This tool can be also used to develop a new architecture, compare with the current one and still helps in change impact analysis, among others features. The architecture comparison can also occur between distinct software systems. The papers show how the *SAVE* tool has been successfully applied to the Common Ground software, a shared software architecture used by NASA missions software systems, in order to avoid further SA maintenance and evolution problems [35] [36]. However, the paper **S07** [35] reports in more detail the workshops that exposed the results found by the *SAVE* tool in the Common Ground's architecture analysis and evolution. Despite the presentation of the tool resources and features along the studies, these papers do not discuss its limitations.

The paper **S09** [37] presents *Motive*, a prototype of an alternative approach to comprehension of software and its architecture evolution, which in addition to showing the evolution or changes of entities or components filtered by period and level of abstraction - like most of existing visual tools for SA evolution - allows users to visualize the net effect on the SA of any set of logically related changes. This set is called *change-set*. Two java open source systems were used to study and evaluate the *Motive* tool and this alternative approach. The authors [37] report that this evaluation showed that the identification and visualization of the impact of *change-sets* seems very promising to help architects and developers comprehend the evolution of a software system and its architecture.

The paper **S10** [38] introduces *YARN* (*Yet Another Reverse-engineering Narrative*), a prototype tool that implements an approach to modeling, extracting and animating a SA evolution. Animating the changing dependencies is an intuitive and natural way to visually realize, identify and compare changes over system lifespan. The *YARN* generates animation through the *YARN balls*. The *YARN ball* is a type of circle composed by subsystems (vertices) and changing dependencies (edges), as shown in Figure 5. The animation starts by showing a *YARN ball* at a chosen baseline and then showing all the sequences of releases of a *YARN ball* progressively. Colour and thickness of edges varying according to the number of changes and how many dependencies exist between two modules. One of the suggested uses of a *YARN ball* by the authors [38] is to help the communication between stakeholders of change based dependency information in software projects. The authors also report that there was created an informal user survey to evaluate the usefulness of the solution and the understanding of the software and architecture evolution.

The paper **S11** [39] presents the description and goals of the project titled “*Development of a methodology, software-suite and service for supporting software architecture recon-*

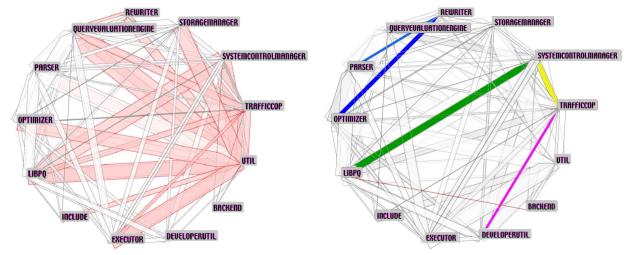


Figure 5: Shots of *YARN Balls* [38]

struction”, intended to develop a methodology and a tool-set (environment) to do automatic architecture reconstruction of software systems through visual resources utilization. It also provides tracking of changes in architectural components during software evolution. At the time this paper was written (2010) the project was focused to systems that has been built using Java or .NET technologies and deal with SQL databases. This paper presents limitation of its study, but only in a summarized way. It also shows details of the current status (2010) of this project [39].

The paper **S12** [40] introduces *Evolve*, a graphical modeling tool that implements an ADL (*Architectural Description Language*) named *Backbone* that is focused on software architecture evolution. *Evolve* supports definition and evolution of SA using the *Backbone*, with particular attention to incremental change and unplanned change processing, very common activities in the software development and evolution process. *Backbone* provides constructs that allow changes that may result in architectural anomalies but *Evolve* is able to detect these anomalies. The paper shows the main characteristics of *Evolve* and how it deal with changes definition in SA, besides a brief use historic. The *Evolve* tool, at the time the paper was published (2011), was freely available for academic research and the production of open source software under the *GNU Affero General Public License version 3*.

The paper **S13** [41] presents and proposes the development of *ARAMIS* (*Architecture Analysis and Monitoring Infrastructure*), an architecture meta-model based solution to extract run-time architecture information and provide data to generate new dynamic architecture views in real time. The solution provides visual representations of the monitored architecture at several abstraction levels, as well as the availability of methods to evaluate this architecture [41]. This study does not present any type of implementation (only technical description), despite presents some limitations of the solution.

The paper **S14** [42] introduces *eCITY*, a tool that helps software architects and developers to understand the software structure of their system. It allows the track of components' insertion, removal, or modification over system lifespan and provides an interactive visualization that provides an overview of changes. All of this implemented under a city metaphor using animations to represent the transitions of the architecture components and color coding to highlight the evolution and changes of these components (Figure 6). The

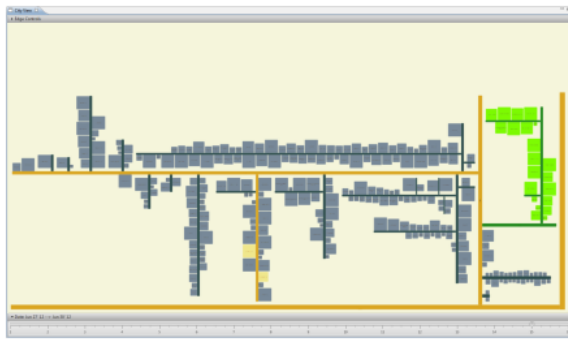


Figure 6: *eCITY*: a City View [42]

eCITY provides an overview of the entire system at a desired point into its evolution process (life cycle), implementing it under a city metaphor, allowing the user an interactive way to understand and explore these changes. *eCITY* provides views to help the changes over time, like: *Timeline View*, an administrative view that uses charts and color to emphasize changes between software system versions; and *City View*, a city layout using animations to represent the transitions of the architecture components and color coding to highlight the evolution and changes of these components, as shown in Figure 6. The *eCITY* tool works with compile-time information, not providing dynamic views [42]. The *eCITY* was originally designed as an Eclipse plug-in. This paper also presents a summary of a conducted user study to emphasize its usefulness.

The paper S15 [43] describes the implementation results of some core characteristics of *ARAMIS*, previously proposed in S07 [41]. *ARAMIS* is a approach for evolution and evaluation of software systems that relies on a infrastructure of run-time monitoring to manage the behavior of the system, in several abstraction levels. In this paper, a prototype of *ARAMIS* was developed focused only in reconstruction of object-level interactions. The prototype uses aspect-oriented techniques to extract and gather the run-time architectural information, and the *XMPP* (*Extensible Messaging and Presence Protocol*) to distribute the gathered information for visualization in real-time, through specialized components, as we can see in Figure 7. The evaluation of this process, according to prototype results, shows that *ARAMIS* can easily be used to demonstrate the behavior of the run-time monitored systems [43].

The paper S16 [44] evaluates the *eCITY* tool, presented in S14 [42]. The authors designed and conducted a controlled experiment to perform the evaluation. In this experiment, they proposed *eCITY* as a tool for improving the analysis of SA evolution along its lifespan. They hoped that through the use of this tool, the architects would be more efficient and effective when perform the analysis of architectural changes. As a result of this experiment, the participants obtained an average gain of 170% in efficiency and an average gain of 15% in the effectiveness of the basic tasks of SA evolution. The authors [44] considered that efficiency means the time required for accomplishing a set of given tasks and ef-

[h!]

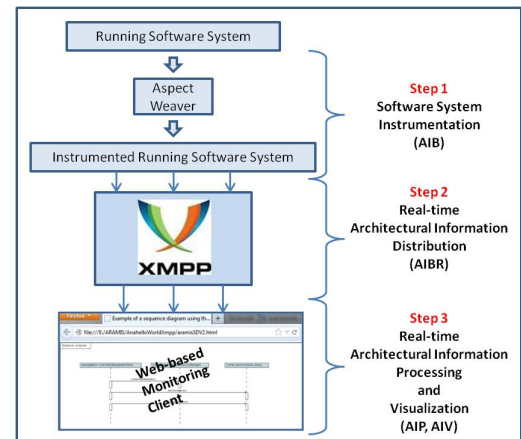


Figure 7: *ARAMIS*: a Prototype Overview [43]

fectiveness means the difference between the true and actual score related to a task.

The paper S17 [45] presents *eCITY+*, an improved version of *eCITY*, presented in S14 [42], now combining the stable city layout and the *Hierarchical Edge Bundling* (*HEB*), an useful technique to help the implementation of 3D visualization with the use of animation. The *eCITY+* is a later version of *eCITY* tool presented in S08 [42], owning characteristics similar from its predecessor. The *eCITY+* tool primarily differs from its earlier version in the use of *HEB* to highlight changes in both the hierarchical structure as well as the inter-dependencies between software components. The *eCITY+* tool, as its predecessor, performs analysis of software architecture relationships through interactive visual support, using the city metaphor to provide an overview of the entire system [45]. The *eCITY+* also was consciously developed as a plug-in to traditional SA maintenance tools.

The paper S18 [46] updates the current status of the *ARAMIS*, previously presented in S13 [41] and S15[43], presenting it now for understanding, communication integrity validation and evaluation of the behavior view of a software architecture. This paper analysis a J2EE application to exemplify how *ARAMIS* can automatically validates the communication between the units of a software system, verifying if it corresponds to its architecture model, including making visualizations of these interactions available on higher and more comprehensible abstraction levels [46].

The paper S19 [47] describes a technical solution to modernize monolithic applications into microservices using software visualization to support the comprehension of evolution process. This conceptual solution can provide a modernization process that uses a legacy system and generates a set of visual diagrams that help architects and developers to understand the system, also suggesting ways of code partitions for transforming into micro-services. The paper has focus only in an understanding stage of modernization, not in transformation stage [47]. The authors analyzed a large Java EE application to validate this solution.

The paper S20 [48] presents a graphical approach that

combines a functional decomposition analysis technique with a non-functional impact analysis technique. Theoretically the functional decomposition prevents the architecture from degrading, but sometimes can be too expensive to implement it. This approach intend to avoid this problem. The functional decomposition technique uses individual relations (associations and attributes) as atomic units of decomposition, partitioning them between the subsystems according to how they are used by the system operations [49]. This technique facilitates the selection of decompositions of low coupling and high cohesion. The non-functional impact analysis technique uses the *KAMP (Karlsruhe Architectural Maintainability Prediction)* approach [50]. The combination of this two techniques guarantee a good equilibrium between functional modularity and non-functional concerns. Finally, this approach is illustrated with an example of evolution of a hypothetical system [48].

The paper S21 [3] introduces *EVA (Evolution Visualization for Architectures)*, a visual tool to help software architects understand the evolution of architecture and therefore track and analyze architectural changes. This tool can visualize and explore architectures of software systems with a long life cycle, including stages of its evolution. *EVA* provides three main views: *Single-Release Architecture View*, that shows the architecture of only one software system version, as shown in Figure 8a; *3-D Architecture-Evolution View*, that depicts architectures of multiple software system versions in a single compositional view, as shown in Figure 8b; and *Pairwise Architecture-Comparison View*, that presents the architectural differences between two software system versions, as shown in Figure 8c. *EVA* allows its users to assess the impact of design decisions, as well as its rationale, which have influenced in the software architecture. As we can see in Figure 8, *EVA* uses color coding to distinct packages or groups of code level entities. The work is currently (2018) focused on showing the explicit reasons behind the architectural changes, in order to assist the rationale of tracking design during the software lifespan [3]. This article presents limitations of its study, but does not discuss them explicitly. The *EVA* tool was developed in Python and is available in a GitHub repository.

Unfortunately, we could not find any taxonomy to classify solutions that support the comprehension of SA evolution using visual resources. Then, to improve the understanding of these visual solutions previously described, we propose a taxonomy with the goal to classify their main characteristics, properties and features. Next, we present the taxonomy and the corresponding classification of each of the selected visual solutions in this SMS.

5.1. Taxonomy for Visual Solutions to Support SA Evolution Comprehension

In this subsection, we present the resulting taxonomy of visual solutions to support the comprehension of software architecture evolution. We argue that the concepts presented in this taxonomy are key to understand the characteristics of visual solutions and therefore to answer the research questions

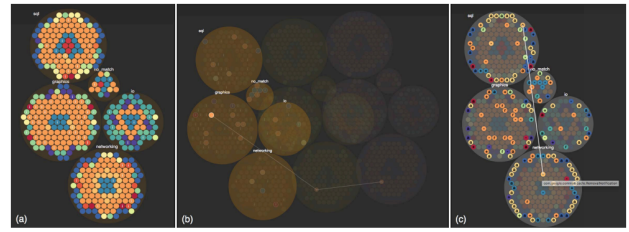


Figure 8: Three Types of Visualization of EVA [3]

of this secondary study. We used the software visualization taxonomy proposed by Price, Baecker and Small (1993) [12] and the framework purposed by Gallagher, Hatch and Munro (2008) [22] as references. The first focused on software visualization, whereas the latter focused on software architecture. For this reason, we adopted them as references to propose the new taxonomy. According to Sulr et al. [26], a taxonomy is consisted of a number of dimensions (e.g., “visualization form”) with their attributes (e.g., “2D Elements”, “Color Coding”). Each visual solution from the selected papers from this SMS can pertain to one or more attributes from a specific dimension, as we will describe in the following paragraphs.

The **Category** dimension is related to the type of proposed visual solution to support the comprehension of SA evolution. It is comprised of four attributes: *Description*, *Technique*, *Tool* and *Environment*. The *Description* attribute classifies the solution or its core idea as a main concept or a technical description of a visual solution. The *Technique* attribute informs whether the visual solution proposes the use of a technique, such as specialized procedures or processes. The *Tool* attribute means the solution presents or proposes a tool, or its development, to assist specialized users of software architecture to develop, or maintain software systems. The *Environment* attribute indicates that the solution explicitly presents or proposes an environment, i.e., an integrated set of tools to help software architecture specialized users in the development or maintenance of software systems. The value of this dimension for each solution is mandatory and it may have more than one attribute signalized.

The **Stage** dimension has four attributes. The *Conceptual* attribute indicates that the solution is still represented and referenced as a *concept*, not having any implementation. The *Project* attribute indicates that the solution has been under development as a project. The *Prototype* attribute means the solution is a prototype of the solution. Finally, the *Stable Release* attribute is related to the status of the solution as already in production as a stable release. The value of this dimension is also mandatory and only one attribute can be signalized for each solution.

The **Visualization Form** dimension defines the visual characteristics of the output of the solution [12]. It is characterized by eight attributes as follows. The *2D Elements* attribute indicates that the solution uses 2D elements, such as 2D charts, diagrams, shapes, windows, figures and lines. The *3D Visualization* attribute indicates that the solution uses

3D resources for visualization. The *Animation* attribute means the solution uses resources of animation in the visual representations. The *Bigraphs* attribute marks the use of bigraphs in the visual solution. The *Visual Metaphor* attribute indicates the use of one or more visual metaphors in the visual representations to enhance the SA evolution understanding. The *Color Coding* attribute means the solution adopts a specific color system to represent the data. The *Tree-based* attribute is used to characterize solutions that use visual structures based on trees. The *UML-based* indicates that the solution uses UML diagrams as a form of visual representation.

The **Static Representation** dimension shows what architectural information can be extracted and represented before run-time [22]. It has two associated attributes: *Static Visualization* and *Recovery*. The *Static Visualization* attribute means the solution displays data exclusively related to static structure of the software system. The *Recovery* attribute indicates that the solution supports the retrieval of architectural data from specific sources. This dimension may have no attributes flagged.

The **Dynamic Representation** dimension shows what architectural information can be extracted and represented during run-time [22]. The *Dynamic Visualization* attribute means the solution displays data extracted during its execution (run-time). The *Events Monitoring* indicates if the solution perform the catch events during its execution. These events can be identified and associated with SA elements and thereby support the comprehension of specific scenarios of software architecture evolution. The *Live* attribute points out that the SA data is gathered in a *real time* fashion as the solution is executed [12] [22]. The *Post-mortem* attribute means the SA data to be gathered is produced in a *post-mortem* fashion by the solution, i.e., generated by its previous execution [12] [22]. This dimension may have no attributes flagged.

The **Architectural Tasks** dimension is related with features of the visual solution that support stakeholders to perform tasks that to some extent focuses on the software architecture and its evolution [22]. It has nine attributes as follows. The *Anomalies* attribute indicates that the solution supports the identification of anomalies, violations and inconsistencies occurrences related to SA. These occurrences can also influence the *Comprehension* attribute indicates the solution supports visual analysis tasks to improve the comprehension of SA and its evolution. Analysis tasks means tasks that generate results to facilitate the understanding the SA, its components, dependencies and relationships, as well as its evolution. They should support top-down or bottom-up approaches [22]. The *Styles* attribute indicates that the solution is able to identify architectural styles and/or verify its compliance with a predefined reference. The *Show Evolution* attribute indicates that the solution provides facilities to exhibit evolution evidence of a SA, in a basic or advanced way [22]. The *Construction* attribute indicates that the solution provides resources to add, change or remove SA elements in the visual representation. The *Evaluation* attribute means the solution supports SA quality analysis and also compliance evaluation. The *Comparison* attribute points out

that the solution performs visual comparison among releases of the software system under analysis. A typical use of this attribute is the comparison between the *as-is* with the *to-be* architectures or the *as-designed* with *as-implemented* software architecture [22]. The *Tracking* indicates that the solution supports the tracking of SA changes throughout its releases. This is a key resource to, for example, identify and trace the architectural decay of a SA, which impairs the software lifespan [3]. Finally, the *Rationale* indicates that the solution presents and make available the rationale behind the design decisions that somehow influences the SA.

5.2. Visual Solutions According to Taxonomy

The Table 11 shows the main characteristics, properties and features identified in the visual solutions of the selected papers from the perspective of the proposed taxonomy. These characteristics were identified and collected exclusively based on the text provided by the selected studies listed in Table 10.

The column labeled *Category* indicates different categories of solutions found in the selected papers, according the description presented in Subsection 5.1. The column labeled *Stage* means the stage of the solution proposed by the paper at the time it was published. The column labeled *Visualization Form* means the summary description of fundamental characteristics related to what can be exhibited in the visual solution. The content of this column is based on the *Form* category proposed by the taxonomy of Price [12]. The columns labeled *Static Representation*, *Dynamic Representation*, *Architectural Tasks* are based on keys areas proposed in Gallagher's framework [22]. The column *Others Features* shows complementary purposes, features and characteristics of the presented visual solution not listed before.

We decided to present a analysis about the visual solution named *EVA* to illustrate how the characteristics of the selected visual solutions presented in Table 11 can be determined. This analysis is shown in Table 12. The visual solution *EVA* was previously presented in the paper S21 [3] and its justifications come exclusively from this paper's content.

The Figure 9 presents the overview of current stages for all visual solutions referenced by Table 11. Its worth remembering that current stages means the stage at the time its study was published. Note that most of solutions (9 in 21) were in "Stable Release" stage, whereas few of them (2 in 21) were in "Project" stage.

The Figure 10 shows the categories of solution found in selected papers. The "Tool" category has the major preference in visual solutions, being present in 14 of 21 solutions found. The "Description" category is present in only three solutions.

Figure 11 shows that the tasks *Comprehension* and *Show Evolution* are adopted in all visual solutions from the selected studies. It means that all of them reported the support of analysis tasks to improve the SA comprehension. Moreover, they also reported the adoption of facilities to exhibit the SA evolution. These are in fact, minimum requirements of a visual solutions to support comprehension of SA evolution. The task *Comparison* is also representative in the an-

Table 11
Using the Proposed Taxonomy to Classify the Visual Solutions to SA Evolution

Ref. Paper	Name of Visual Solution	Category	Stage	Visualization Form	Static Representation	Dynamic Representation	Architectural Tasks	Other Features
S01	Beagle	Environment, Technique	Prototype	2D Elements, Tree-based	Static Visualization, Recovery	N/A	Comprehension, Comparison, Show Evolution	Evolution metrics usage
S02	Retrospective Analysis	Technique	Prototype	2D Elements, Color Coding	Static Visualization, Recovery	N/A	Comprehension, Comparison, Show Evolution, Evaluation	N/A
S03	Not named	Technique	Conceptual	2D Elements, Visual Metaphor	Static Visualization	N/A	Comprehension, Construction, Show Evolution, Styles	Graph Grammar, Synchronized Hyperedge Replacement Systems, Hyperedge Replacement Grammar
S04	ABACUS	Tool	Stable Release	2D Elements, Tree-based, 3D Visualization, Color Coding	Static Visualization, Recovery	N/A	Comprehension, Construction, Show Evolution, Evaluation	Methodology-oriented
S05	Not named	Description, Technique	Conceptual	2D Elements, Bigraph	Static Visualization	Dynamic Visualization, Live	Comprehension, Comparison, Construction, Show Evolution, Styles	BRS resources
S06	Film Strip and Dependency Matrix	Technique	Prototype	2D Elements	Static Visualization, Recovery	N/A	Comprehension, Comparison, Show Evolution	N/A
S07	SAVE	Tool, Environment	Stable Release	2D Elements	Static Visualization, Recovery	N/A	Comprehension, Anomalies, Comparison, Construction, Show Evolution, Rationale, Styles, Evaluation	Process-oriented, Products comparison
S08	SAVE	Tool, Environment	Stable Release	2D Elements	Static Visualization, Recovery	N/A	Comprehension, Anomalies, Comparison, Construction, Show Evolution, Rationale, Styles, Evaluation	Process-oriented, Products comparison
S09	Motive	Tool	Prototype	2D Elements, UML-based	Static Visualization, Recovery	N/A	Comprehension, Show Evolution	N/A
S10	YARN	Tool	Prototype	2D Elements, Color Coding, Animation	Static Visualization, Recovery	N/A	Comprehension, Show Evolution	N/A
S11	GOP	Tool, Environment	Project	2D Elements, Color Coding	Static Visualization, Recovery	N/A	Comprehension, Comparison, Construction, Show Evolution, Tracking	Methodology-oriented
S12	Evolve	Tool	Stable Release	2D Elements, UML-based, Animation	Static Visualization	N/A	Comprehension, Anomalies, Comparison, Construction, Show Evolution	Model-driven, ADL implementation
S13	ARAMIS	Tool	Conceptual	N/A	Static Visualization	Dynamic Visualization, Events Monitoring	Comprehension, Show Evolution, Evaluation	Model-driven
S14	eCITY	Tool	Stable Release	2D Elements, Color Coding, Animation, Visual Metaphor	Static Visualization	N/A	Comprehension, Comparison, Show Evolution, Tracking	N/A
S15	ARAMIS	Tool, Technique	Prototype	2D Elements, UML-based	N/A	Dynamic Visualization, Events Monitoring, Live, Post-mortem	Comprehension, Show Evolution, Evaluation	Model-driven, Traceability with requirements, Views creation
S16	eCITY	Tool	Stable Release	2D Elements, Color Coding, Animation, Visual Metaphor	Static Visualization	N/A	Comprehension, Comparison, Show Evolution, Tracking	N/A
S17	eCITY+	Tool, Technique	Stable Release	2D Elements, 3D Visualization, Color Coding, Animation, Visual Metaphor	Static Visualization	N/A	Comprehension, Comparison, Show Evolution, Tracking	As-plugin
S18	ARAMIS	Tool	Stable Release	2D Elements, UML-based	N/A	Dynamic Visualization, Events Monitoring, Live, Post-mortem	Comprehension, Show Evolution, Evaluation	Model-driven, Traceability with requirements, Views creation, Communication integrity validation
S19	Not named	Description	Project	2D Elements, UML-based, Color Coding	Static Visualization	N/A	Comprehension, Comparison, Show Evolution	Model-driven, Modernization
S20	Not named	Technique	Conceptual	2D Elements, UML-based	Static Visualization	N/A	Comprehension, Show Evolution	KAMP approach
S21	EVA	Tool	Stable Release	2D Elements, 3D Visualization, Color Coding	Static Visualization, Recovery	N/A	Comprehension, Comparison, Show Evolution, Tracking, Rationale	ADD traceability

alyzed solutions. They have reported the ability to perform visual comparison of SA characteristics among two or more releases of a specific software system, which corresponds to 62% of analyzed visual solutions. The tasks *Anomalies*

and *Rationale* were not representative in the solutions, corresponding to only 14% and close to *Styles* with 19%. The task *Construction* explicitly appears in 33% of the visual solutions and half of them present specific resources to design

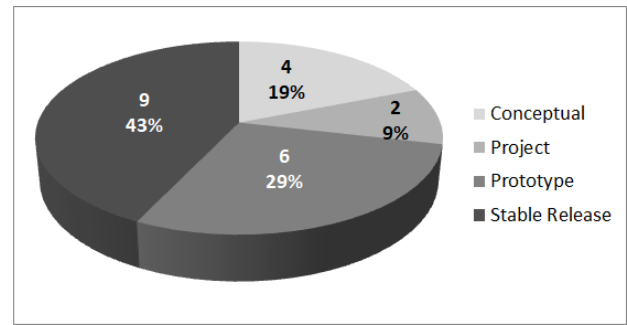
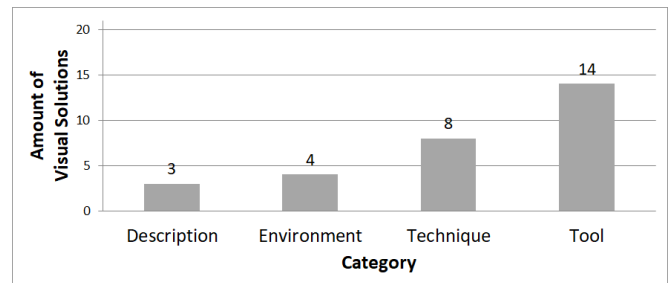
Table 12
Sample Analysis - Visual Solution EVA [3]

Values	Justifications
Category = ("Tool")	Title of the paper is "EVA: A Tool for Visualizing Software Architectural Evolution".
Stage = ("Stable Released")	The behavior of EVA presented in the paper always suggests that it is in production; EVA was already evaluated in use and is in the process of deploying to a large development organization.
Visualization Form = ("2D Elements", "3D Visualization", "Color Coding")	"2D Elements": suggested by figures displayed in the paper; "3D Visualization": EVA provides a view that shows architectures of multiple releases in a 3D visualization; "Color Coding": EVA differs code-level entities through color coding.
Static Representation = ("Static Visualization", "Recovery")	"Static Visualization": EVA only collects data at compile-time (static elements); "Recovery": EVA supports many SA recovery techniques.
Dynamic Representation = N/A	The paper does not present any characteristic or feature for dynamic representation.
Architectural Tasks = ("Comprehension", "Comparison", "Show Evolution", "Tracking", "Rationale")	"Comprehension": EVA provides a view that represents a single SA release, allowing users to interactively understand the functionality of each architecture component; "Comparison": EVA provides comparison view that shows the SA differences between two releases; "Show Evolution": EVA visualizes the SA evolution through a set of source codes across multiple releases; "Tracking": EVA provides a view that allows representation of change tracking of entities across multiple releases; "Rationale": EVA collects relevant data of design decisions from system repositories, displaying them together with the architecture evolution visualization.
Other Features = ("ADD Traceability")	EVA shows the traceability of architecture design decisions over time

the architecture, such as S02 solution [34] (using *Bi-graphical Reactive System*) and S06 solution [40] (using an ADL visual modelling component). The task *Evaluation* also appears with 33%.

6. Discussion

The specific research question SRQ1 is related to the main visual solutions to support the software architecture evolution comprehension. The answer is presented in Table 11, where it is possible to identify their main goals and characteristics of each solution. The specific research question SRQ2 is concerned to different purposes of using the visual solutions to support the software architecture evolution comprehension. The purposes are also presented in Table 11, identified primarily in the *Architectural Tasks* col-

**Figure 9:** Current Stages of Visual Solutions**Figure 10:** Categories of Visual Solutions

umn and also in the *Static Representation*, *Dynamic Representation* and *Other Features* columns. The specific research question SRQ3 focuses on solutions designed to visually support comprehension of software architecture evolution can be classified. The solutions can be classified in the categories *Description*, *Technique*, *Tool* and *Environment*. The Table 11 classify each solution and shows that the *Tool* category has more representants among the selected studies. The specific research question SRQ4 focuses on visual forms used to support comprehension of software architecture evolution. The Table 11 lists in the *Visualization Form* column how the visual forms have been adopted in the visual solutions discussed in the selected studies. From the list, it is possible to identify the following distribution of use of the visual forms: 2D Elements (20 studies), Tree-based (2 studies), Color coding (9 studies), Visual metaphor (4 studies), 3D Visualization (3 studies), Bi-graph (1 study), Uml-based (5 studies) and Animation (5 studies).

Finally, the main research question (RQ) focuses on the evaluation of the usage of visual solutions to support the comprehension of software architecture evolution based on papers published in the peer-reviewed literature. Table 11 presents an up-to-date overview of solutions used for the stated purpose with different characteristics and strategies as has been already explained for the specific research questions. We have identified that based on evidence from the selected studies, features of comprehension and SA evolution visualization are minimum requirements to support software architecture evolution comprehension, as shown in Figure 11. On the other hand, the low number of papers found in

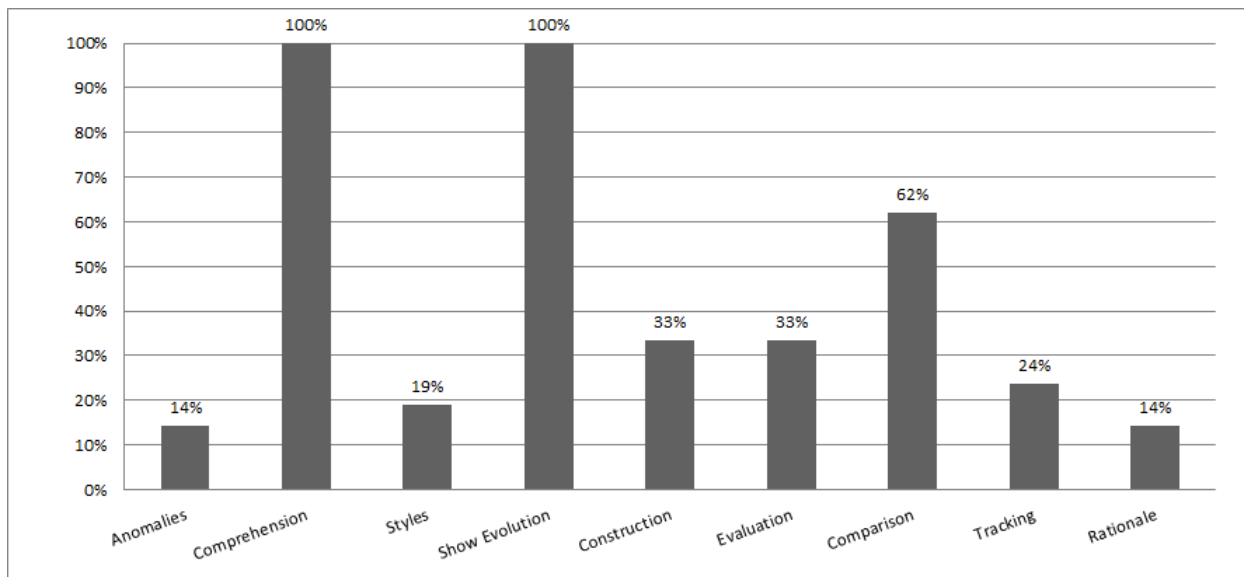


Figure 11: Distribution of Architectural Tasks in the Selected Papers

this systematic mapping study, suggests that visual solutions to support SA evolution comprehension is an area that needs expand in terms of studies and options available for practitioners and researchers.

7. Conclusion

The aim of this work is to report the design, execution and results of a systematic mapping study of visual solutions to support the comprehension of software architecture evolution. We performed a SMS according to the plan described in Section 4. Initially, the applied search strings retrieved 211 papers from the selected electronic databases. This number was reduced to 21 after applying all selection procedures criteria. From these 21 selected studies, the identified visual solutions to support the comprehension of SA evolution were classified as follows: 14% categorized as *Description*, 38% as *Technique*, 67% as *Tool* and 19% as *Environment*. All of them support the architectural tasks *Comprehension* and *Show Evolution*.

Besides the identification of visual solutions from the literature, another contribution of this study is a taxonomy to classify these solutions. The taxonomy contains six dimensions: category, stage, visualization form, static representation, dynamic representation and architectural tasks. These dimensions and their attributes was explained in Section 5, SubSection 5.1. Each visual solution was classified according to this taxonomy, generating a table of characterization of visual solutions to SA evolution, presented in Table 11. The assignment of the taxonomy attributes value to the visual solution characteristics shows a current overview of the available visual solutions in peer-reviewed literature.

Moreover, this study also shows that visual solutions to support SA evolution comprehension usually present features to support analysis tasks to improve the SA comprehension and also provide facilities to exhibit the SA evolution.

This study also concludes that, due to a few number of papers found in this SMS, the studies may consider to allocate more research and development effort to provide effective visual solutions to support SA evolution comprehension, improving the acquired knowledge in this area.

As a future work, we recommend extending the research to establish a methodology or process, based on the taxonomy proposed, to define projects to build visual solutions of SA evolution comprehension. Another possibility for future work is the improvement of the proposed taxonomy, aiming to generate a new framework to objectively evaluate solutions like that ones discussed in this SMS.

References

- [1] H. P. Breivold, I. Crnkovic, M. Larsson, A systematic review of software architecture evolution research, *Information and Software Technology* 54 (2012) 16 – 40.
- [2] L. Yu, S. Ramaswamy, J. Bush, Symbiosis and software evolvability, *IT Professional* 10 (2008) 56–62.
- [3] D. Nam, Y. K. Lee, N. Medvidovic, Eva: A tool for visualizing software architectural evolution, in: 2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion), 2018, pp. 53–56.
- [4] Q. Tu, M. W. Godfrey, An integrated approach for studying architectural evolution, in: *Proceedings 10th International Workshop on Program Comprehension*, 2002, pp. 127–136.
- [5] M. Lungu, M. Lanza, Exploring inter-module relationships in evolving software systems, in: 11th European Conference on Software Maintenance and Reengineering (CSMR'07), 2007, pp. 91–102.
- [6] R. Taylor, N. Medvidovic, E. Dashofy, *Software Architecture: Foundations, Theory and Practice*, Hoboken, New Jersey, 2009.
- [7] D. Garlan, J. M. Barnes, B. Schmerl, O. Celiku, Evolution styles: Foundations and tool support for software architecture evolution, in: 2009 Joint Working IEEE/IFIP Conference on Software Architecture European Conference on Software Architecture, 2009, pp. 131–140.
- [8] M. Shahin, P. Liang, M. A. Babar, A systematic review of software architecture visualization techniques, *Journal of Systems and Software* 94 (2014) 161 – 185.
- [9] M. Shahin, P. Liang, M. R. Khayyambashi, Improving understand-

- ability of architecture design through visualization of architectural design decision, in: *Proceedings of the 2010 ICSE Workshop on Sharing and Reusing Architectural Knowledge, SHARK '10*, 2010, pp. 88–95.
- [10] R. L. Novais, M. G. de Mendonca Neto, *Computer Systems and Software Engineering: Concepts, Methodologies, Tools, and Applications*, Chapter: Software Evolution Visualization: Status, Challenges, and Research Directions, IGI Global, 2018.
- [11] Y. Ghanam, S. Carpendale, A survey paper on software architecture visualization Technical report (2008).
- [12] B. A. Price, R. M. Baecker, I. S. Small, A principled taxonomy of software visualization, *Journal of Visual Languages Computing* 4 (1993) 211–266.
- [13] J. Werther, G. Carneiro, R. Maciel, A systematic mapping on visual solutions to support the comprehension of software architecture evolution, in: *Proceedings of the 25th International DMS Conference on Visualization and Visual Languages, DMSVIVA '19*, 2019, pp. 63–74. doi:10.18293/DMSVIVA2019-008.
- [14] D. Garlan, D. E. Perry, Introduction to the special issue on software architecture, *IEEE Transactions on Software Engineering* 21 (1995) 269–274.
- [15] D. Garlan, Software architecture: A travelogue, in: *Proceedings of the on Future of Software Engineering, FOSE 2014*, 2014, pp. 29–39.
- [16] D. Garlan, Software architecture: a roadmap, in: *Proceedings of Conference on the Future of Software Engineering, Limerick, Ireland*, 2000, pp. 91–101.
- [17] N. Medvidovic, R. Taylor, D. Rosenblum, An architecture-based approach to software evolution, in: *Proceedings of International Workshop on the Principles of Software Evolution*, 1998.
- [18] L. A. Belady, M. M. Lehman, A model of large program development, *IBM Systems journal* 15 (1976) 225–252.
- [19] C. F. Kemerer, S. Slaughter, An empirical approach to studying software evolution, *IEEE Transactions on Software Engineering* 25 (1999) 493–509.
- [20] D. Rowe, J. Leane, D. Lowe, Defining systems evolvability - a taxonomy of change, in: *Proceedings of International Conference and Workshops on Engineering of Computer-Based Systems (ECBS)*, 1998, Jerusalem, Israel, 1998.
- [21] E. Gamma, et al., *Design patterns: elements of reusable object-oriented software*, Addison-Wesley, 1995.
- [22] K. Gallagher, A. Hatch, M. Munro, Software architecture visualization: An evaluation framework and its application, *IEEE Transactions on Software Engineering* 34 (2008) 260–270.
- [23] J. Cleland-Huang, R. S. Hanmer, S. Supakkul, M. Mirakhorli, The twin peaks of requirements and architecture, *IEEE Software* 30 (2013) 24–29.
- [24] V. Basili, G. Caldiera, H. Rombach, The goal question metric paradigm, *Encyclopedia of Software Engineering* 2 (1994) 528–532.
- [25] A. Telea, L. Voinea, H. Sassenburg, Visual tools for software architecture understanding: A stakeholder perspective, *IEEE Software* 27 (2010) 46–53.
- [26] M. Sulir, M. Bacikova, S. Chodarev, J. Poruban, Visual augmentation of source code editors: A systematic mapping study, *Journal of Visual Languages Computing* 49 (2018) 46–59.
- [27] C. Wohlin, et al., *Experimentation in Software Engineering*, Springer-Verlag, 2012.
- [28] V. R. Basili, H. D. Rombach, The tame project: towards improvement-oriented software environments, *IEEE Transactions on Software Engineering* 14 (1988) 758–773.
- [29] T. Dyba, T. Dingsoyr, Empirical studies of agile software development: A systematic review, *Information and Software Technology* 50 (2008) 833–859.
- [30] D. Moher, A. Liberati, J. Tetzlaff, D. G. Altman, P. Group, et al., Preferred reporting items for systematic reviews and meta-analyses: the prisma statement, *PLoS medicine* 6 (2009) e1000097.
- [31] M. Jazayeri, On architectural stability and evolution, in: J. Blieberger, A. Strohmeier (Eds.), *Reliable Software Technologies — Ada-Europe 2002*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 13–23.
- [32] D. Hirsch, U. Montanari, Two graph-based techniques for software architecture reconfiguration, *Electronic Notes in Theoretical Computer Science* 51 (2002) 177–190.
- [33] K. Dunsire, T. O'Neill, M. Denford, J. Leane, The abacus architectural approach to computer-based system and enterprise evolution, in: *12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05)*, 2005, pp. 62–69. doi:10.1109/ECBS.2005.66.
- [34] Z. Chang, X. Mao, Z. Qi, An approach based on bigraphical reactive systems to check architectural instance conforming to its style, in: *First Joint IEEE/IFIP Symposium on Theoretical Aspects of Software Engineering (TASE '07)*, 2007, pp. 57–66.
- [35] W. C. Stratton, D. E. Sibol, M. Lindvall, P. Costa, Technology infusion of save into the ground software development process for nasa missions at jhu/apl, in: *2007 IEEE Aerospace Conference*, 2007, pp. 1–15. doi:10.1109/AERO.2007.352763.
- [36] W. C. Stratton, D. E. Sibol, M. Lindvall, P. Costa, The save tool and process applied to ground software development at jhu/apl: An experience report on technology infusion, in: *31st IEEE Software Engineering Workshop (SEW 2007)*, 2007, pp. 187–193.
- [37] A. McNair, D. M. German, J. Weber-Jahnke, Visualizing software architecture evolution using change-sets, in: *14th Working Conference on Reverse Engineering (WCRE 2007)*, IEEE, 2007, pp. 130–139.
- [38] A. Hindle, Z. M. Jiang, W. Kuleilat, M. W. Godfrey, R. C. Holt, Yarn: Animating software evolution, in: *2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, IEEE, 2007, pp. 129–136.
- [39] L. Schrettnner, P. Hegedus, R. Ferenc, L. J. Fulop, T. Bakota, Development of a methodology, software – suite and service for supporting software architecture reconstruction, in: *2010 14th European Conference on Software Maintenance and Reengineering*, 2010, pp. 190–193.
- [40] A. McVeigh, J. Kramer, J. Magee, Evolve: Tool support for architecture evolution, in: *Proceedings of the 33rd International Conference on Software Engineering, ICSE '11*, 2011, pp. 1040–1042.
- [41] A. Dragomir, H. Lichter, Model-based software architecture evolution and evaluation, in: *2012 19th Asia-Pacific Software Engineering Conference*, volume 1, 2012, pp. 697–700.
- [42] T. Khan, H. Barthel, A. Ebert, P. Liggesmeyer, ecity: A tool to track software structural changes using an evolving city, in: *2013 IEEE International Conference on Software Maintenance*, 2013, pp. 492–495. doi:10.1109/ICSM.2013.80.
- [43] A. Dragomir, H. Lichter, Run-time monitoring and real-time visualization of software architectures, in: *2013 20th Asia-Pacific Software Engineering Conference (APSEC)*, volume 1, 2013, pp. 396–403.
- [44] T. Khan, H. Barthel, L. Guzman, A. Ebert, P. Liggesmeyer, ecity: Evolutionary software architecture visualization—an evaluation, in: *Building Bridges: HCI, Visualization, and Non-formal Modeling*, Springer, 2014, pp. 201–224.
- [45] T. Khan, S. R. Humayoun, K. Amrhein, H. Barthel, A. Ebert, P. Liggesmeyer, ecity+: A tool to analyze software architectural relations through interactive visual support, in: *Proceedings of the 2014 European Conference on Software Architecture Workshops, ECSAW '14*, 2014, pp. 36:1–36:4.
- [46] A. Nicolaescu, H. Lichter, A. Göringer, P. Alexander, D. Le, The aramis workbench for monitoring, analysis and visualization of architectures based on run-time interactions, in: *Proceedings of the 2015 European Conference on Software Architecture Workshops, ACM*, 2015, p. 57.
- [47] D. Escobar, D. Cárdenas, R. Amarillo, E. Castro, K. García's, C. Parra, R. Casallas, Towards the understanding and evolution of monolithic applications as microservices, in: *2016 XLII Latin American Computing Conference (CLEI)*, 2016, pp. 1–11.
- [48] D. Faitelson, R. Heinrich, S. S. Tyszbewicz, Supporting software architecture evolution by functional decomposition., in: *MODEL-SWARD*, 2017, pp. 435–442.
- [49] D. Faitelson, S. Tyszbewicz, Improving design decomposition, in:

International Symposium on Dependable Software Engineering: Theories, Tools, and Applications, Springer, 2015, pp. 185–200.

- [50] K. Rostami, J. Stammel, R. Heinrich, R. Reussner, Architecture-based assessment and planning of change requests, in: Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures, ACM, 2015, pp. 21–30.

Journal of Visual Language and Computing

Volume 2019, Number 1