VII CONGRESSO BRASILEIRO DE SOFTWARE · TEORIA E PRÁTICA

CBSOFT

MARINGÁ 2016

X Workshop em Desenvolvimento Distribuído de Software, Ecossistemas de Software e Sistemas-de-Sistemas

CBSOFT.ORG

REALIZAÇÃO: SBC Sociedade Brasileira de Computação

EXECUÇÃO: UEM Universidade Estadual de Maringá · UTFPR Universidade Tecnológica Federal do Paraná Câmpus Campo Mourão · din Departamento de Informática

ORGANIZAÇÃO: TASA Eventos

APOIO: Convention & Visitors Bureau Maringá e Região

FOMENTO: CNPq Conselho Nacional de Desenvolvimento Científico e Tecnológico · Paraná Governo do Estado Secretaria da Ciência, Tecnologia e Ensino Superior · Fundação Araucária Apoio ao Desenvolvimento Científico e Tecnológico do Paraná · UEM Universidade Estadual de Maringá · CAPES

PATROCINADORES GIGA: UniCesumar · COLIVRE Cooperativa de Tecnologias Livres

PATRCCINADORES MEGA: ESPWEB Especialização em Web e Mobile · ThoughtWorks®

VII CONGRESSO BRASILEIRO DE SOFTWARE • TEORIA E PRÁTICA

CBSOFT

MARINGÁ 2016

X WORKSHOP DE DESENVOLVIMENTO DISTRIBUÍDO DE SOFTWARE, ECOSSISTEMAS DE SOFTWARE E SISTEMAS-DE-SISTEMAS (WDES 2016)

21 de setembro de 2016 | *September 21, 2016*
Maringá, PR, *Brazil*

ANAIS | *PROCEEDINGS*

Sociedade Brasileira de Computação – SBC

REALIZAÇÃO | *REALIZATION*
Sociedade Brasileira de Computação (SBC)

EXECUÇÃO | *EXECUTION*
Universidade Estadual de Maringá (UEM) – Departamento de Informática (DIN)
Universidade Tecnológica Federal do Paraná (UTFPR) – Câmpus Campo Mourão (UTFPR-CM)

ISBN: 978-85-7669-337-6

Apresentação

O Workshop em Desenvolvimento Distribuído de Software, Ecossistemas de Software e Sistemas-de-Sistemas (WDES 2016), em sua décima edição, traz como principal tema os "Desafios da Sustentabilidade e da Diversidade no Desenvolvimento de Sistemas de Software". O WDES constitui um fórum para apresentação e discussão de resultados e experiências de pesquisadores e praticantes das áreas de Desenvolvimento Distribuído de Software (DDS), Ecossistemas de Software (ECOS) e Sistemas-de-Sistemas (SoS), visando a geração de conhecimento que possa viabilizar projetos de sucesso nessas três áreas e/ou nas suas relações. Além disso, o workshop visa ampliar as possibilidades de colaboração em âmbito nacional e internacional, bem como consolidar as pesquisas em DDS, ECOS e SoS como uma área estratégica em Engenharia de Software no Brasil, como ocorre no exterior.

O Comitê Diretivo do WDES 2016 é constituído por três pesquisadores de cada uma das áreas envolvidas no workshop. Por sua vez, o Comitê de Programa do WDES 2016 é formado por 55 pesquisadores de instituições do Brasil e do exterior, com atuação e produção relevantes nas áreas de pesquisa envolvidas no workshop. Os membros do Comitê de Programa conduziram um processo rigoroso de revisão, sendo que cada artigo foi avaliado por pelo menos três membros.

O WDES 2016 recebeu 24 submissões, sendo 22 artigos válidos (9 em inglês e 13 em português). Após o processo de revisão do Comitê de Programa, além da análise final do Comitê Diretivo, foram aceitos 7 artigos completos e 3 artigos curtos. Haverá premiação do melhor artigo e convite para submissão de versão estendida para uma edição especial de periódico. Para estimular a discussão de pesquisas com potencial, 4 trabalhos serão ainda apresentados como pôsteres. Por fim, o workshop vai contar com uma dinâmica para a organização da Agenda de Pesquisa e de Colaboração da Comunidade WDES.

É com satisfação que damos as boas-vindas aos autores e apresentadores de artigos, da academia e da indústria. Também recebemos com grande prazer os demais participantes do CBSoft 2016, que gostaríamos de convidar a tomar parte ativamente das discussões e momentos de integração proporcionados pelo workshop. Adicionalmente, gostaríamos de agradecer a todos os demais autores que submeteram seus artigos, aos membros do Comitê de Programa e do Comitê Diretivo, e aos organizadores e patrocinadores do CBSoft 2016 pelo suporte na realização deste workshop.

Esta edição é organizada conjuntamente pela Universidade Federal de Pernambuco (UFPE) e Softex Recife, Universidade Federal do Amazonas (UFAM) e Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (ICMC/USP). O evento é realizado no Centro Universitário Unicesumar, em Maringá, Paraná, Brasil, no dia 21 de setembro, em conjunto com o VII Congresso Brasileiro de Software: Teoria e Prática (CBSoft 2016). Esperamos que tenham uma ótima estada em Maringá!

Maringá, setembro de 2016.

Ivaldir Honório de Farias Junior (SOFTEX RECIFE - UFPE)
Arilo Claudio Dias Neto (UFAM)
José Carlos Maldonado (ICMC/USP)

Foreword

The Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (WDES 2016), in its tenth edition, aims at putting together competencies and technologies of these three related areas: Distributed Software Development (DSD), Software Ecosystems (SECO), and Systems-of-Systems (SoS). This year's main topic is "Challenges of Sustainability and Diversity in the Development of Software Systems". This workshop consists of a forum for presentation and discussion of results and experiences of researchers and practitioners of DSD, SECO, and SoS. As its main goal, the workshop intends to generate knowledge that makes it possible to conduct successful projects in these areas. Besides, the workshop tries to leverage opportunities for national (and international) collaborations, as well as strengthen researches on DSD, SECO, and SoS as a strategic area in the Brazilian Software Engineering, as abroad.

The WDES 2016 Steering Committee is composed of three researchers from each workshop area. For its part, the WDES Program Committee is composed of 55 researchers from national and international institutions, with relevant expertise and production in the related research areas of the workshop. Members of the Program Committee conducted a rigorous review process, in which each article was assessed by at least three members.

In this WDES edition, we received 24 submissions being 22 valid papers (9 papers in English and 13 in Portuguese). After the reviewing process from the Program Committee and the final analysis of the Steering Committee, 7 full papers and 3 short papers were accepted. There will be an award for the best paper and an invitation to submit an extended version to a special issue of a journal. Additionally, 4 papers will be presented as posters, since they discuss potential research ideas. Finally, we will promote a Special Session to define the WDES Research and Collaboration Roadmap.

We welcome the authors and articles presenters with delight, from both academia and industry. We also welcome with great pleasure the CBSoft 2016 participants, which we would like to invite to actively take part in discussions and integration moments provided by the workshop. Additionally, we would like to thank all the other authors who submitted their articles, the Steering and Program Committee members, and the organizers and sponsors of CBSoft 2016, for their support for the accomplishment of this workshop.

This edition is jointly organized by the Federal University of Pernambuco (UFPE) and Softex Recife, the Federal University of Amazonas (UFAM) and the Institute of Mathematics and Computer Science of the University of São Paulo (ICMC - USP). The event is held in the University Center of Maringá, in Maringa, State of Parana, Brazil, on September 21st, co-located with VII Brazilian Congress on Software: Theory and Practice (CBSoft 2016).

We wish a pleasant stay in Maringá!

Maringá - Paraná, September 2016.

Ivaldir Honório de Farias Junior (SOFTEX RECIFE - UFPE)
Arilo Claudio Dias Neto (UFAM)
José Carlos Maldonado (ICMC/USP)

Comitê técnico | *Technical committee*

Coordenadores de comitê de programa | *PC chairs*

Ivaldir Honório de Farias Junior (SOFTEX RECIFE - UFPE)
Arilo Claudio Dias Neto (UFAM)
José Carlos Maldonado (ICMC/USP)

Comitê diretivo | *Steering committee*

Arilo Claudio Dias Neto (UFAM)
Cláudia Werner (COPPE/UFRJ)
Elisa Yumi Nakagawa (ICMC/USP)
Flavio Oquendo (IRISA - UBS)
Heitor Costa (UFLA)
Ivaldir de Farias Junior (SOFTEX RECIFE - UFPE)
José Carlos Maldonado (ICMC/USP)
Rodrigo Santos (UNIRIO)
Sabrina Marczak (PUCRS)

Comitê de programa | *Program committee*

Adriano Albuquerque (UNIFOR)
Alexandre L'Erario (UTFPR)
Aline Vasconcelos (IFF)
Andrea Magdaleno (UFF)
Carina Alves (UFPE)
Carlos Cuesta (King Juan Carlos University)
Christina Chavez (UFBA)
Cláudia Werner (COPPE/UFRJ)
Clodis Boscarioli (UNIOESTE)
Cristiano Maciel (UFMT)
Daniela Cruzes (SINTEF)
Davi Viana (UFMA)
Elias Genvigir (UTFPR)
Elisa Huzita (UEM)
Elisa Yumi Nakagawa (ICMC/USP)
Felipe Furtado (CESAR.Edu)
Fernanda Campos (UFJF)
Fernando Figueira Filho (UFRN)
Flavia Delicato (DCC/UFRJ)
Flávio Oquendo (IRISA - UBS)

Revisores externos | *External reviewers*

Página Web e suporte | *Website and support*

Artigos técnicos | *Technical papers*

Pôsteres | *Posters*

Towards a Model for Task Allocation in Distributed Software Development

Marum Simão Filho[1,2], Plácido R. Pinheiro[2], Adriano B. Albuquerque[2]

[1]Faculdade 7 de Setembro – FA7 – Fortaleza, CE – Brasil

[2]Universidade de Fortaleza – UNIFOR – Fortaleza, CE – Brasil

{marumsimao,placidrp}@gmail.com,{marum,plácido,adrianoba}@unifor.br

Abstract. *The management of distributed software development projects presents many challenges. One of them consists of the allocation of tasks between remote teams. When allocating a task, the project manager takes into account factors such as technical expertise and time zone. The project manager usually makes this decision in a subjective way. The verbal decision analysis is an approach based on solving problems through multi-criteria qualitative analysis. This paper describes the progress of studies towards the definition of a model for task allocation in projects of distributed software development based on verbal decision analysis methods ORCLASS and ZAPROS III-i.*

1. Introduction

The development of systems is a business that is still growing a lot on the world scenario. Historical players of the computing area, such as IBM, Oracle, SAP and Microsoft, are increasingly consolidated. With the advent of the Internet and the Web, other companies established and consolidated as giants. In this case, Google and Facebook appear as the best examples. These consolidated companies attend to different segments of activity, from industry to trade, from financial system to power generation, from agriculture to civil construction, from entertainment to education. The types of systems are also as diverse as possible: integrated management systems, operating systems, among others.

Time requirements have become more demanding, in other words, clients cannot expect 2 to 3 years for a system or application any more. This dynamic business environment requires organizations to develop and evolve software at much higher speeds than in the past. New methodologies, as the agile ones, arose and brought a new way of thinking about building software, focusing mainly on the result for the customer, i.e. the customer product delivery.

Large software companies often have offices in several different cities, or even, in different countries, to serve customers around the world. The teams in the various offices can have different profiles or expertise. Some teams perform tasks that require proximity to customer, but other tasks can be done remotely. Some kinds of activities can be carried out away from the big cities, where labor is often more expensive. Thus, the companies take advantage of the cheap work force.

Software development consists of several different activities, such as requirements elicitation, analysis and design, coding, testing, among others. Organizations that work with well-designed projects and adopt consistent methodologies can distribute these activities among different sites, benefiting from the best skills of each local team. In this scenario, a company that has remote offices can distribute its work packages in several ways.

In this context, the project manager must decide which task to allocate to each remote team based on various criteria. This is a decision-making problem with a high degree of subjectivity, which is appropriate for verbal decision analysis. Considering the given scenario, this paper describes a proposal of a model to aid task allocation in projects of distributed software development based on the verbal decision analysis methods ORCLASS and ZAPROS III-i. Preliminary researches and initial results are presented.

The remainder of the paper is organized as follows: Section 2 shortly presents issues involving task allocation in software distributed development. Section 3 provides a brief description of the verbal decision analysis methods ORCLASS and ZAPROS III-i. Section 4 describes the preliminary researches towards the elaboration of the allocation model. Section 5 presents some initial results. Section 6 introduces the proposed model. Finally, in Section 6, we provide the conclusions and suggestions for further work.

2. Task Allocation in Distributed Software Development

The allocation of tasks is a critical activity for any kind of project, especially in a distributed scenario. Most of the time, few factors drive the allocation of tasks, such as hand labor costs. Risks and other relevant factors such as the workforce skills, innovation potential of different regions, or cultural factors are often insufficiently recognized (Lamersdorf, Münch and Rombach, 2008).

Many studies about task allocation in Distributed Software Development (DSD) have been carried out along the years aiming at mapping this topic and its features. Lamersdorf, Münch and Rombach (2008) developed an analysis of the existing approaches to distribution of duties that involved procedures for the distributed development, distributed generation, and distributed systems areas. Lamersdorf, Münch and Rombach (2009) conducted a survey on the state of practice in DSD in which they investigated the criteria that influence task allocation decisions. Lamersdorf and Münch (2010) presented TAMRI (Task Allocation based on Multiple cRIteria), a model based on multiple criteria and influencing factors to support the systematic decision of task allocation in distributed development projects.

Ruano-Mayoral et al. (2013) presented a methodological framework to allocate work packages among participants in global software development projects. Marques et al. (2011) performed a systematic mapping, which enabled to identify models that propose to solve the problems of task allocation in DSD projects. They intended to propose a combinatorial optimization-based model involving classical task scheduling problems. Marques, Rodrigues and Conte (2012) performed a tertiary review applying the systematic review method on systematic reviews that address DSD issues on task allocation.

Galviņa and Šmite (2011) provided an extensive literature review for understanding the industrial practice of software development processes and concluded that the evidence of how these projects are organized is scarce. Babar and Zahedi (2012) presented a literature review considering the studies published in the International Conference in Global Software Engineering (ICGSE) between 2007 and 2011. They found that the vast majority of the evaluated studies were in software development governance and its sub-categories, and much of the work had focused on the human aspects of the GSD rather than technical aspects.

Almeida, Albuquerque and Pinheiro (2011) presented a multi-criteria decision model for planning and fine-tuning such project plans: Multi-criteria Decision Analysis (MCDA). The model was developed using cognitive mapping and MACBETH (Measuring Attractiveness by a Categorical Based Evaluation Technique) (Bana e Costa et al. (2011). In Almeida, Albuquerque and Pinheiro (2011a), they applied (MCDA) on the choice of DSD Scrum project plans that have a better chance of success. Simão Filho, Pinheiro and Albuquerque (2015) conducted a quasi-systematic review of studies of task allocation in DSD projects that incorporate agile practices. The study brought together a number of other works, allowing the establishment of the many factors that influence the allocation of tasks in DSD.

3. Verbal Decision Analysis

Decision-making is an activity that is part of people's and organizations' lives. In most problems, to make a decision, a situation is assessed against a set of characteristics or attributes, i.e., it involves the analysis of several factors, also called criteria. When a decision can generate a considerable impact, such as management decisions, and must take into account some factors, the use of methodologies to support the decision-making process is suggested, because choosing the inappropriate alternative can lead to waste of resources, time, and money, affecting the company.

The decision-making scenario that involves the analysis of alternatives from several viewpoints is called multi-criteria decision analysis and is supported by multi-criteria methodologies (Bana e Costa, 1992). These methodologies favor the generation of knowledge about the decision context, which helps raise the confidence of the decision maker (Evangelou, Karacapilidis, and Khaled, 2005). The Verbal Decision Analysis (VDA) is an approach to solving multi-criteria problems through qualitative analysis (Larichev and Brown, 2000). VDA supports the decision-making process through the verbal representation of problems. VDA methodologies can be used for ordering or sorting the alternatives. Among the classification methods, we can mention ORCLASS, SAC, DIFCLASS, and CYCLE. Some sorting methods are PACOM, ARACE, and those from the ZAPROS family (ZAPROS-LM, STEPZAPROS, ZAPROS III and III-i) (Tamanini, 2014).

3.1. The ORCLASS Method for Classification

ORCLASS methodology aims at classifying the alternatives in a given set: the decision maker needs these alternatives to be categorized into a small number of decision classes or groups, usually two. The first group covers the most preferable alternatives, and the less preferable alternatives belong to the second one (Larichev and Moshkovich, 1997).

The flowchart to apply the ORCLASS method was presented in (Tamanini, 2010). In that scheme, the application of the ORCLASS method can be divided into three stages. In the first stage, the problem's formulation, the set of criteria, criteria values, and the decision groups are defined. Then, the construction of the classification rule is carried out based on the decision maker's preferences. We use the same concepts presented in (Larichev and Moshkovich, 1997), based on which a classification task is presented as a set of boards. Each cell of the board is composed of a combination of values for each criterion defined for the problem, which represents a possible alternative to the problem (Machado, 2012). Finally, the results are generated and analyzed, i.e., the alternatives are classified into two groups, the preferable and the not preferable ones.

3.2. The ZAPROS III-i Method for Rank Ordering

The ZAPROS methodology aims at ranking multi-criteria alternatives in scenarios involving a rather small set of criteria and criteria values, and a great number of alternatives (Larichev, 2001). ZAPROS-III-i method's flowchart to rank order a set of alternatives can be found in (Tamanini, 2010), which is divided into three stages: Problem Formulation, Elicitation of Preferences and Comparison of Alternatives.

In the first stage, the relevant criteria and their respective values to the decision-making process are obtained. In the second stage, the scale of preferences is generated based on the decision maker’s preference. In the last stage, the method performs the comparison between the alternatives based on the decision maker’s preferences. The method carries out the elicitation of preferences by comparing vectors of alternatives (Tamanini and Pinheiro, 2008).

4. Preliminary Researches

To classify and rank order the most important factors that project managers should take into account when allocating tasks in projects of distributed development of software, we applied a hybrid methodology, which consists of five main steps, as explained in the next subsections.

A - Identification of the Influencing Factors

First, we conducted a literature research to identify the main influencing factors that should be considered when allocating tasks in projects of distributed development of software. Table 1 shows the factors found as result of this research, and that worked as the alternatives to our decision problem (Simão Filho, Pinheiro and Albuquerque, 2015).

Table 1. Influencing Factors on Task Allocation in DSD Projects

ID	Alternatives
Factor1	Technical expertise
Factor2	Expertise in business
Factor3	Project manager maturity
Factor4	Proximity to client
Factor5	Low turnover rate
Factor6	Availability
Factor7	Site maturity
Factor8	Personal trust
Factor9	Time zone
Factor10	Cultural similarities
Factor11	Willingness at site

B - Definition of the Criteria and Criteria values

Next, we interviewed a group of 4 project management experts to define the criteria and the criteria values. This is the definition stage of the criteria. For each criterion, we established a scale of values associated with it (Machado et al., 2010). The criteria values were ordered from the most preferable value to the least preferable one.

As result of this step, we got the list of criteria and criteria values for the problem of selecting the most important factors to be considered in task allocation in DSD projects, which is listed on Table 2 (Simão Filho, Pinheiro and Albuquerque, 2016c).

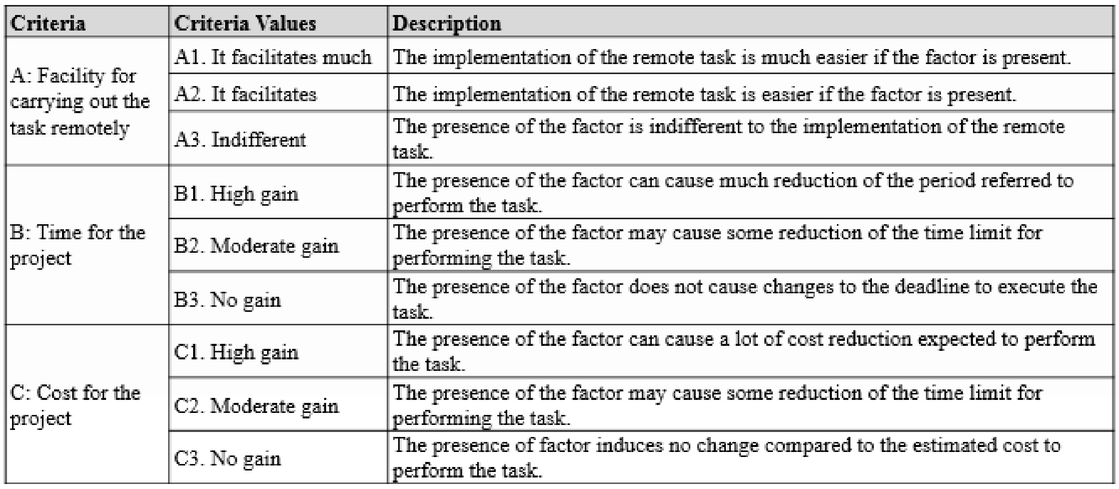
C - Definition of Alternatives, Decision Groups, and Alternatives’ Characterization

We created a questionnaire to gather information and opinions about the factors that influence the allocation of tasks in DSD projects. We applied the questionnaire over the Web to a group of 20 project managers and consisted of two sections. The first section aimed to trace the respondents profile about his/her professional experience and education (Simão Filho, Pinheiro and Albuquerque, 2016c).

The respondents’ profiles can be summarized as follows. 30% of respondents have bachelor's degree, 60% have master's degrees and 10% are doctors or higher. All respondents work with software development for over 8 years. 40% work in private companies and 60% in public companies. 65% work in companies whose business is to provide IT services whereas 35% do not. 40% of respondents have between 4 and 8 years of experience in managing software development projects whereas 60% have more than 8 years. 65% have some certification in project management while 35% do not have any certification. 80% of respondents managed more than 8 software development projects and 20% managed less than 8 projects. Considering projects of distributed development of software, 20% of respondents managed more than 8 projects, 55% managed from 1 to 8 projects, and 25% did not manage any project.

The second section of the questionnaire inquired the views of experts on the factors that influence the allocation of tasks in DSD projects (the influencing factors shown in Table 1). For our problem, we described such influencing factors as alternatives. Thus, in every question, the professional analyzed the influencing factors about a set of criteria and criteria values (shown in Table 2) and selected what criterion value that best fitted the factor analyzed.

Table 2. List of criteria and criteria values with description



Criteria	Criteria Values	Description
A: Facility for carrying out the task remotely	A1. It facilitates much	The implementation of the remote task is much easier if the factor is present.
	A2. It facilitates	The implementation of the remote task is easier if the factor is present.
	A3. Indifferent	The presence of the factor is indifferent to the implementation of the remote task.
B: Time for the project	B1. High gain	The presence of the factor can cause much reduction of the period referred to perform the task.
	B2. Moderate gain	The presence of the factor may cause some reduction of the time limit for performing the task.
	B3. No gain	The presence of the factor does not cause changes to the deadline to execute the task.
C: Cost for the project	C1. High gain	The presence of the factor can cause a lot of cost reduction expected to perform the task.
	C2. Moderate gain	The presence of the factor may cause some reduction of the time limit for performing the task.
	C3. No gain	The presence of factor induces no change compared to the estimated cost to perform the task.

Then, we analyzed the responses to determine the criteria values representing the alternatives. For each influencing factor, we filled the final table based on the replies of the majority of professionals. We then selected the value of the criterion that had the greatest number of choices to represent the alternative. Thus, the decision groups were defined as follows. Group I: The influencing factors that will be selected as the most important ones that project managers should take into account when allocating tasks to remote teams (the preferable factors). Group II: The influencing factors that should be less considered by project managers when they need to allocate tasks to remote teams (not preferable factors).

D - The ORCLASS Method Application

We used the ORCLASSWEB tool to aid the application of the ORCLASS method. It is divided into four steps: Criteria and criteria value definition; Alternatives definition; Construction of the classification rule; and Results Generation (Machado, Pinheiro and Tamanini, 2014). In this step, we had the support of an experienced project manager to answer the questions generated by the ORCLASS method to classify the alternatives. The classification rule was completed based on the decision-maker choices. In the end, the tool processed the full classification of the alternatives.

As result of applying the ORCLASS method, we got the following factors to compose the Group I (the preferable factors): Factor1 - Technical expertise, Factor2 - Expertise in business, Factor3 - Project manager maturity, Factor4 - Proximity to client, Factor5 - Low turnover rate, Factor6 - Availability, Factor7 - Site maturity, Factor11 - Willingness at site, and Factor8 - Personal trust. They are the most important ones that project managers should consider when allocating tasks in projects of distributed development of software. In the Group II, which was composed of the least preferable factors, we got the following factors: Factor9 - Time zone and Factor10 - Cultural similarities (Simão Filho, Pinheiro and Albuquerque, 2016c).

E - The ZAPROS-III-i Method Application

After determining the preferred factors using the OSCLASS method, we moved on to the ordering stage. At this stage, we applied the ZAPROS III-i method to put in order the preferable factors, such that it is possible to establish a ranking of preferred factors. In this step, the least preferable factors were discarded, thereby reducing our workspace (Simão Filho, Pinheiro and Albuquerque, 2016a).

To facilitate the decision-making process and perform it consistently, we used the ARANAÚ tool, presented in (Tamanini, 2007). The use of ZAPROS III-i method in the ARANAÚ tool requires four steps. First, we introduced the criteria presented in the problem into the ARANAÚ tool. Next, the decision-maker decides the preferences. The tool presents questionings that can be easily answered by the decision-maker to obtain the scale of preferences.

The process occurs in two stages: elicitation of preferences for quality variation of the same criteria and elicitation of preferences between pairs of criteria. The questions provided require a comparison considering the two reference situations (Tamanini, 2010). Once the scale of preferences is structured, the next step is to define the problem's alternatives. The alternatives to our problem are the preferable factors integrating of Group I.

4.1. Initial Results

After introducing all the data and answering the necessary questions, the decision maker is presented with the result in a table containing the alternatives and their criteria evaluations, formal index of quality and rank, as exposed in Table 3. Note that there are five alternatives (factors) that are in the same ranking position (first position), and their FIQ's values are equals to zero. This occurs because all of them got the best evaluation according to the survey filled out by the professionals (A1, B1, C1), which is the best possible evaluation (Simão Filho, Pinheiro and Albuquerque, 2016b).

Table 3. The final ranking of alternatives

Rank	Alternative	Representation	FIQ
1	Factor1 - Technical expertise	A1B1C1	0
1	Factor2 - Expertise in business	A1B1C1	0
1	Factor5 - Low turnover rate	A1B1C1	0
1	Factor6 - Availability	A1B1C1	0
1	Factor11 - Willingness at site	A1B1C1	0
2	Factor7 - Site maturity	A1B1C2	6
3	Factor4 - Proximity to client	A1B2C2	10
4	Factor3 - Project manager maturity	A2B2C2	11
5	Factor8 - Personal Trust	A2B2C2	11

5. A Proposal of a Model for Task Allocation in DSD Projects Based on VDA

Based on preliminary researches, we propose a model to aid the task allocation in projects of distributed software development. The model is founded on multi-criteria methods of VDA: ORCLASS for classification, and ZAPROS III-I for rank ordering. The proposed model is structured into four activities: (1) problem characterization, (2) rank ordering the preferable influencing factors (for each kind of software development work package), (3) definition of the scoring rules for remote teams, and (4) construction of the comparison method among remote teams.

In order to achieve a more precise and less general result, we decided to replicate the process of determining the influencing factors for the main groups of tasks in a traditional software development process. We intend to take some work packages (also known as "workflows") from RUP (Rational Unified Process) as starting point, since it is a well known process and widely used throughout the world. Because the following subjects are more likely to be developed remotely, they were chosen to compose the model: requirements, analysis and design, implementation and testing (Kruchten, 2004).

Figure 1 shows a flowchart for the proposed model. The problem characterization activity is divided into three tasks: identification of the influencing factors; definition of the criteria and criteria values; and definition and characterization of the alternatives and definition of the decision groups, for each type of software development task package. In the first task, we will conducted a literature research to identify the main influencing factors that should be considered when allocating tasks in DSD projects. Next, we will interview a group of project management experts to define the criteria and the criteria values in order to assess the influencing factors. Then, we will apply a questionnaire to gather expert's opinions about the factors that influence task allocation in DSD projects. We intend to apply the questionnaire over the Web to project managers of various companies so that we can define and characterize the alternatives, besides defining the decision groups, for each type of software development task package (requirements, analysis and design, implementation and testing), according to VDA methodologies.

In the second activity, rank ordering the preferable influencing factors, first we will classify the influencing factors into preference groups, according to ORCLASS method. Then, we will rank order the influencing factors of the preferable group, according to ZAPROS III-i method. These tasks will be applied for each type of software development task package.

After that, we will move to the next activity, i.e., the definition of the scoring rules for the influencing factors for each location. This activity is still under research. Finally, the construction of the method to allow the comparison among remote teams will be developed. This activity is also under research. In the end, for each type of work package, we will be able to compare the influencing factors among the various locations and thus choose the location that has the best rate. The team at this place should be allocated to develop the work package.

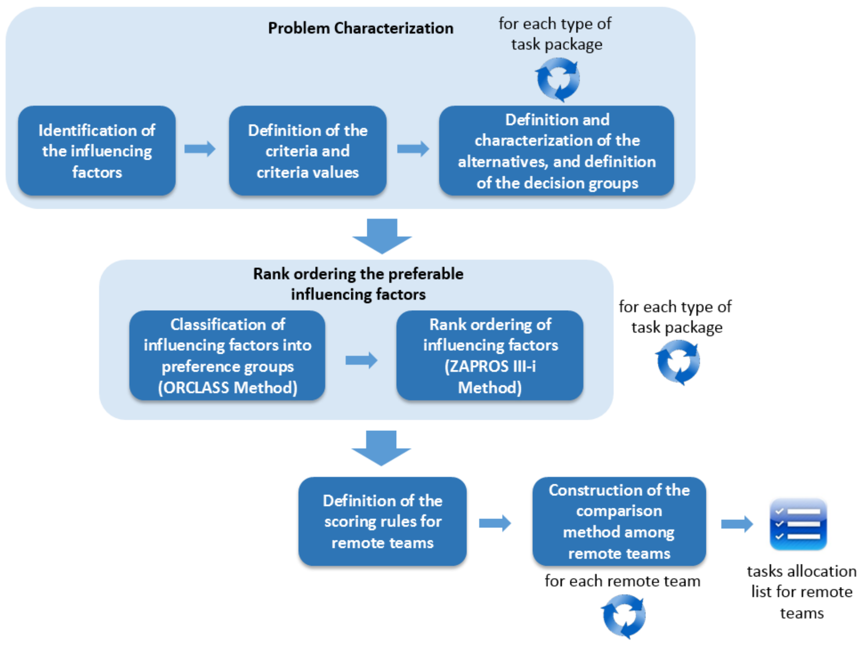

Problem Characterization
for each type of task package
Identification of the influencing factors
Definition of the criteria and criteria values
Definition and characterization of the alternatives, and definition of the decision groups
Rank ordering the preferable influencing factors
Classification of influencing factors into preference groups (ORCLASS Method)
Rank ordering of influencing factors (ZAPROS III-i Method)
for each type of task package
Definition of the scoring rules for remote teams
Construction of the comparison method among remote teams
for each remote team
tasks allocation list for remote teams

Figure 1. Flowchart for the Proposed Model

6. Conclusion and Future Works

The main contribution of this work was to describe the proposal of a model to aid task allocation in projects of distributed development software based on VDA. The proposal involves applying a hybrid methodology based on ORCLASS and ZAPROS III-i methods of VDA framework to classify and rank order the most important factors that project managers should consider when allocating tasks among distributed teams. Preliminary researches were developed which allowed us to present some initial results.

As future work, we intend to classify the influencing factors into preference groups for each kind of work package based on RUP disciplines. We also need to rank order the influencing factors for each kind of work package based on RUP disciplines. In addition, we plan to define the scoring rules for the factors on each site to allow the comparison among the remote teams. Finally, we intend to apply the proposed model in some real projects, and analyze the results in order to check the efficacy of the model.

Acknowledgment

The first author is thankful for the support given by the Coordination for the Improvement of Higher Level-or Education- Personnel (CAPES) and 7 de Setembro College. The second author is grateful to National Counsel of Technological and Scientific Development (CNPq) via Grants #305844/2011-3. The authors would like to thank The Edson Queiroz Foundation/University of Fortaleza for all the support.

References

Almeida, L. H., Albuquerque, A. B. and Pinheiro, P. R. (2011) "A Multi-criteria Model for Planning and Fine-Tuning Distributed Scrum Projects." 6th IEEE International Conference on Global Software Engineering.

—. (2011) "Applying Multi-Criteria Decision Analysis to Global Software Development with Scrum Project Planning." Lecture Notes in Computer Science 6954: 311-320.

Babar, M. A. and M. Zahedi. (2012) "Global Software Development: A Review of the State-Of-The-Art (2007 – 2011)." IT University Technical Report Series.

Bana e Costa, C. A. (1992) "Structuration, Construction et Exploitation Dún Modèle Multicritère D'aide à la Décision." Universidade Técnica de Lisboa.

Bana e Costa, C. A., Sanchez-Lopez, R., Vansnick, J. C. and De Corte, J. M. (2011) "Introducción a MACBETH." Análisis Multicriterio para la Toma de Decisiones: Métodos y Aplicaciones. México: Plaza y Valdés.

Evangelou, C., Karacapilidis, N. and Khaled, O. A. (2005) "Interweaving knowledge management, argumentation and decision making in a collaborative setting: the KAD ontology model." International Journal of Knowledge and Learning: 130–145.

Galviņa, Z. and Šmite, D. (2011) "Software Development Processes in Globally Distributed Environment." Scientific Papers 770.

Kruchten, Philippe. (2004) “The Rational Unified Process: An Introduction”. 3rd ed. Addison-Wesley.

Lamersdorf, A. and Münch, J. (2010) "A multi-criteria distribution model for global software development projects." The Brazilian Computer Society 2010.

Lamersdorf, A., Münch, J. and Rombach, D. (2009) "A Survey on the State of the Practice in Distributed Software Development: Criteria for Task Allocation." Fourth IEEE International Conference on Global Software Engineering, ICGSE 2009.

—. (2008) "Towards a Multi-Criteria Development Distribution Model: An Analysis of Existing Task Distribution Approaches." IEEE International Conference on Global Software Engineering, ICGSE 2008.

Larichev, O. I. and Moshkovich, H. M. (1997) Verbal Decision Analysis for Unstructured Problems. Boston: Kluwer Academic Publishers.

Larichev, O. I. and Brown, R. (2000) "Numerical and verbal decision analysis: comparison on practical cases." Journal of Multi-criteria Decision Analysis 9: 263–273.

Larichev, O. I. (2001) "Ranking multi-criteria alternatives: The Method ZAPROS III." European Journal of Operational Research: 550–558.

Machado, T. C. S. (2012) “Towards Aided by Multi-criteria Support Methods and Software Development: A Hybrid Model of Verbal Decision Analysis for Selecting Approaches of Project Management”, Master Thesis, Master Program in Applied Computer Sciences, University of Fortaleza. Fortaleza.

Machado, T.C.S., Menezes, A.C., Pinheiro, L.F.R., Tamanini, I., and Pinheiro, P.R. (2010) "Applying verbal decision analysis in selecting prototypes for educational tools." Proceedings of IEEE International Conference on Intelligent Computing and Intelligent Systems. Xiamen, China, pages 531–535.

Machado, T. C. S., Pinheiro, P. R. and Tamanini, I. (2014) "OrclassWeb: A Tool Based on the Classification Methodology ORCLASS from Verbal Decision Analysis Framework." Mathematical Problems in Engineering.

Marques, A. B., Rodrigues, R., Prikladnicki, R. and Conte, T. (2011) "Alocação de Tarefas em Projetos de Desenvolvimento Distribuído de Software: Análise das Soluções Existentes." Anais do II Congresso Brasileiro de Software, V WDDS – Workshop de Desenvolvimento Distribuído de Software. São Paulo.

Marques, A. B., Rodrigues, R. and Conte. T. (2012) "Systematic Literature Reviews in Distributed Software Development: A Tertiary Study." Proceedings of IEEE International Conference on Global Software Engineering, ICGSE 2012. Pages 134-143.

Ruano-Mayoral, M., Casado-Lumbreras, C., Garbarino-Alberti, H. and Misra, S. (2013) "Methodological framework for the allocation of work packages in global software development." Journal of Software: Evolution and Process.

Simão Filho, M., Pinheiro, P. R. and Albuquerque, A. B. (2016) "Applying Verbal Decision Analysis in Distributed Software Development-Rank Ordering the Influencing Factors in Task Allocation." Proceedings of the 11th Iberian Conference on Information Systems and Technologies (CISTI'2016), Gran Canaria, España, Vol. I, pages 205-210.

—. (2016) "Applying Verbal Decision Analysis to Task Allocation in Distributed Software Development". Proceedings of the 28th International Conference on Software Engineering & Knowledge Engineering, San Francisco, pages 402-407. DOI 10.18293/SEKE2016-181.

—. (2015) "Task Allocation Approaches in Distributed Agile Software Development: A Quasi-systematic Review". Proceedings of the 4th Computer Science On-line Conference 2015 (CSOC2015), Software Engineering in Intelligent Systems Series, Zlín, Vol. 3, pages 243-252, , DOI 10.1007/978-3-319-18473-9_24.

—. (2016) "Task Allocation in Distributed Software Development aided by Verbal Decision Analysis". Proceedings of the 5th Computer Science On-line Conference 2016 (CSOC2016), Software Engineering Perspectives and Application in Intelligent Systems Series, Zlín, Vol. 2, pages 127-137, DOI 10.1007/978-3-319-33622-0_12.

Tamanini, I. and Pinheiro P. R. (2008) "Applying a New Approach Methodology with ZAPROS." XL Brazilian Symposium on Operations Research, pages 914-925.

Tamanini, I. (2014) "Hybrid Approaches of Verbal Decision Analysis Methods." Doctor Thesis, Graduate Program in Applied Computer Sciences, University of Fortaleza.

—. (2010) "Improving the ZAPROS Method Considering the Incomparability Cases." Master Thesis, Master Program in Applied Computer Sciences, University of Fortaleza.

—. (2007) "Uma ferramenta Estruturada na Análise Verbal de Decisão Aplicando ZAPROS (A structured tool in Verbal Decision Analysis Applying ZAPROS)." Computer Sciences, University of Fortaleza.

Fatores que Afetam a Comunicação em Projetos Distribuídos de Software e Estratégias de Mitigação: Um Estudo na Indústria Brasileira de TI

Júlia Mara Colleoni Couto[1], **Josiane Kroll**[1]

[1]Faculdade de Informática (FACIN)
Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Av. Ipiranga, 6681 - Partenon, CEP 90619-900 – Porto Alegre – RS – Brasil

{julia.couto, josiane.kroll}@acad.pucrs.br

Abstract. *Distributed Software Development (DDS) has increased been adopted in the Brazilian IT (Information Technology) industry, even it with many challenges. Many these challenges are related to communication fails among teams. Thus, this study aims to identify the main factors that affect communication in DDS projects in the Brazilian IT industry and strategies to mitigate these challenges. Our data collection came from interviews with ten project managers from different IT organizations located in Brazil. As a result, we found seven factors, which affect communication in DDS projects and eleven strategies to reduce communication problems.*

Resumo. *O Desenvolvimento Distribuído de Software (DDS) vem sendo cada vez mais adotado na indústria brasileira de tecnologia de informação (TI), embora ainda existam muitos desafios para sua prática. Muitos destes desafios originam-se de falhas na comunicação entre equipes. Este artigo visa identificar os principais fatores que afetam a comunicação em projetos de DDS na indústria brasileira de TI e estratégias adotadas para a mitigação desses fatores. Os dados deste estudo foram coletados através de entrevistas realizadas com dez gerentes de projetos de diferentes empresas de TI localizadas no Brasil. Como resultado, foram identificados sete fatores que que afetam a comunicação em projetos de DDS e onze estratégias para reduzir problemas de comunicação.*

1. Introdução

O Desenvolvimento Distribuído de Software (DDS) é uma tendência na indústria brasileira de software. Entretanto, ainda existem muitos desafios para a sua prática [Jimenez et al. 2009]. Em projetos que adotam o DDS, o gerenciamento das informações requer atenção especial, pois suas características únicas podem potencializar as probabilidades de falha na comunicação.

Existem evidências de que projetos de DDS podem trazer benefícios para as empresas, como acesso a locais com abundância de mão de obra especializada, redução nos custos e proximidade com os clientes [Sievi-Korte et al. 2015], mas também existem algumas barreiras a serem transpostas para que esses projetos possam ser mais eficientes. Alguns dos principais desafios nesse tipo de projeto remetem à comunicação, consciência de grupo e coordenação das equipes, entre outros [Jimenez et al. 2009].

Falhas na comunicação podem levar ao aumento de custos e mudanças no cronograma [Mishra and Mahanty 2015], o que impacta diretamente no planejamento e na execução do projeto. Quando os membros da equipe não se entendem, as possibilidades de as tarefas não serem executadas ou serem executadas erroneamente são grandes, e dessa maneira o cronograma e o plano do projeto são negativamente impactados.

No presente artigo, são investigados os fatores que afetam a comunicação e estratégias adotadas para reduzir as falhas de comunicação em projetos DDS. Para isto, foram realizadas entrevistas com gerentes de projetos sobre os problemas de comunicação mais frequentes ou impactantes, e como eles lidam com essas questões. Os respondentes fazem parte de uma amostra composta por organizações globais de desenvolvimento de software. Os gerentes de projetos são brasileiros, mas atuam em organizações que possuem sites distribuídos em diferentes países. Por motivos de confidencialidade, não é permitido dar um maior detalhamento dessa amostra.

Para que os resultados da pesquisa fossem alcançados, procedeu-se uma análise qualitativa, onde foram analisadas dez entrevistas, realizadas com gerentes de projetos que atuam ou atuavam em projetos DDS na indústria brasileira. A principal contribuição refere-se a um conjunto de estratégias de mitigação relacionadas aos fatores que afetam a comunicação.

Este artigo está organizado da seguinte forma: na Seção 2 é apresentada a base teórica deste estudo, a qual fornece uma visão geral do DDS e apresenta os trabalhos relacionados. Na Seção 3, é descrita a metodologia de pesquisa empregada neste estudo. Na Seção 4, são apresentados os resultados obtidos, os quais são discutidos na Seção 5. A Seção 6 apresenta as limitações, e a Seção 7 traz as conclusões obtidas neste estudo e identifica as oportunidades para trabalhos futuros.

2. Base Teórica

O PMBOK (*Project Management Body of Knowledge*) diz que o gerenciamento das comunicações é uma das dez áreas essenciais a serem levadas em consideração para o gerenciamento dos projetos [PMI 2013]. Essa área de conhecimento reúne procedimentos para planejamento, coleta, criação, distribuição, armazenamento, recuperação, gerenciamento, controle, monitoramento e disponibilização oportuna e apropriada das informações em um projeto. Gerenciar a comunicação é um processo complexo que deve ser cuidadosamente desenhado e executado de maneira que facilite os processos de planejamento, gerenciamento e controle das informações.

Um projeto DDS permite que os membros sejam alocados em vários sites remotos durante o projeto do software, criando uma rede de sub-times geograficamente dispersos [Jimenez et al. 2009]. De acordo com Dingsoyr e Smite [Dingsoyr and Smite 2014], o DDS continua sendo um desafio para muitas empresas, que ainda estão se esforçando para aumentar a eficácia de seus projetos. Além disso, as características do DDS agregam mais complexidade aos processos de software já existentes [Colomo-Palacios et al. 2014]. Um exemplo disso são os processos que requerem um grande fluxo de comunicação. A comunicação no DDS em escala global se torna difícil, principalmente devido as diferentes línguas, culturas e a distribuição das equipes em diferentes fusos-horários [Calefato et al. 2012]. Por outro lado, outras características do DDS como a distribuição da equipe em diferentes locais de desenvolvimento possibilita a alocação de recursos qualificados para o projeto. Essa característica do DDS é considerada uma vantagem, pois permite que a empresa crie equipes especializadas [Nguyen-Duc et al. 2015].

2.1. Trabalhos Relacionados

Falhas na comunicação com o cliente ou com a equipe remota é um problema que pode ter consequências desastrosas em um projeto de DDS. No estudo conduzido por [Daim et al. 2012], foram identificados cinco fatores que influenciam em falhas na comunicação em equipes geograficamente dispersos. Esses fatores são: confiança, relações interpessoais, diferenças culturais, liderança e tecnologia. Tendo em vista que todos esses fatores requerem atenção redobrada, é de extrema importância que o gerente do projeto exerça a liderança de maneira a direcionar os integrantes da equipe do projeto para que ajam como uma equipe, perseguindo os objetivos do projeto em conjunto.

No estudo realizado por [Anderson et al. 2007], foi investigada a natureza da comunicação em reuniões de equipes virtuais. Foi analisado o quanto, qual o conteúdo e qual o padrão das interações entre as pessoas que se comunicam utilizando ferramentas institucionais, como vídeo-conferências e aplicativos para compartilhamento, por exemplo. Observou-se que o uso dessas tecnologias aumentou significativamente a troca de informações e comunicação entre os membros das equipes distribuídas.

A importância das tecnologias de informação e telecomunicação para estabelecer e manter a confiança em projetos distribuídos é reforçada em [Stawnicza 2015]. Este estudo também mostra como o uso dessas tecnologias colabora para reforçar o sentimento de unidade das equipes. [Weimann et al. 2013] complementa que a má escolha de tecnologias de comunicação ou acesso limitado à internet influenciam no desempenho, satisfação das equipes e reduzem a eficiência na troca de informações entre as equipes.

No estudo realizado por [Paasivaara and Lassenius 2003] é apresentada uma coleção de processos e práticas que podem ser utilizados para minimizar o efeito da dispersão geográfica em equipes distribuídos. Dentre as práticas citadas, três são centradas na comunicação: resolução de desafios, informação e monitoramento e práticas de construção de relacionamentos.

As dimensões psicológicas relacionadas ao sentimento de distância entre os integrantes de equipes distribuídos é alvo do estudo de [Prikladnicki 2012]. Nesse estudo, identificou-se que a comunicação é um dos fatores mais críticos quando se trata de percepção da distância pelas pessoas.

[Herbsleb et al. 2005] introduz a importância do aspecto cultural na comunicação em projetos distribuídos. Algumas diferenças culturais são bastante peculiares, e acabam por gerar barreiras de comunicação nos projetos. Por exemplo, entre a cultura asiática e a cultura europeia e americana, encontra-se diferenças entre a maneira das pessoas ao expressar acordo e desacordo, assim como em questionar quando não se entende determinado tópico. Os americanos e europeus normalmente são mais francos e diretos quando não concordam, e perguntam mais quando não entendem determinado tópico, diferentemente dos asiáticos. Assim, é necessário se certificar de que houve pleno entendimento por parte de seus equipes, para garantir que o trabalho seja feito conforme o planejado.

Em comparação com os trabalhos acima citados, o presente estudo mostra o cenário de comunicação dos gerentes de projetos de DDS que atuam na indústria brasileira de

TI. Como um dos principais diferenciais a ser apresentado, ressalta-se os problemas no contexto da indústria brasileira de software.

3. Metodologia de Pesquisa

Entrevistas foram conduzidas com gerentes de projeto de diferentes organizações com o objetivo de identificar ambientes de comunicação, desafios de comunicação em projetos de DDS e soluções para mitigá-los. Para isso foram definidas três perguntas, as quais guiaram as entrevistas.

1. Você poderia descrever o ambiente de comunicação em projetos DDS?
2. Você poderia descrever os fatores que afetam a comunicação e que levam a crises em projetos DDS?
3. Você poderia descrever estratégias/soluções para mitigar esses fatores que afetam a comunicação em projetos DDS?

Foram coletados dados de dez gerentes de projeto localizados em organizações no Brasil. As entrevistas foram realizadas no período entre outubro de 2014 e março de 2015. O nível de experiência dos participantes é de, em média, seis anos. A maioria dos participantes também possuía experiência prévia como desenvolvedor de software. Na Tabela 1 são apresentadas informações sobre os participantes.

Tabela 1. Detalhes dos participantes entrevistados neste estudo.

Participante	Projetos DDS gerenciados	Experiência na gerência de projetos DDS
1	26	7 anos
2	7	9 anos
3	10	11 anos
4	15	10 anos
5	5	2 anos
6	20	8 anos
7	6	8 anos
8	8	8 anos
9	6	4 anos
10	4	8 anos

Neste estudo foi empregada a análise de conteúdo, que é um tipo de análise qualitativa dos dados, a qual contribuiu para identificar os principais desafios de comunicação como também suas consequências para o desenvolvimento de projetos distribuídos.

4. Resultados

Nesta seção são reportados os resultados deste estudo. Os resultados então organizados nos três principais tópicos investigados: 1) Ambiente de comunicação em projetos de DDS, 2) Fatores que afetam a comunicação, e 3) Estratégias/soluções para mitigar problemas de comunicação.

4.1. Ambiente de Comunicação em Projetos DDS

O ambiente de comunicação em projetos de DDS pode variar de acordo com o número de sites em um projeto. Um vez que os membros da equipe estão distribuídos em diferentes sites, eles adotam diferentes ferramentas de comunicação tais como: e-mail, IM (*Instant Message*) e *conference calls*, incluindo VOIP, *conference bridge* (tipo especializado de equipamento que conecta linhas de telefone) e vídeo conferência.

De acordo com os participantes, o e-mail e o IM são as tecnologias mais utilizadas para comunicação em projetos de DDS. Em segundo lugar aparecem as *conference calls*. Esse tipo de tecnologia é utilizada para discutir e resolver problemas técnicos. Skype e Live Meeting são exemplos de tecnologias que permitem a realização de *conference calls*.

A tecnologia de vídeo conferência, embora recomendada pela literatura [Paasivaara et al. 2008] e reconhecida pelos participantes como uma tecnologia que aproxima os membros da equipe e facilita a comunicação, é adotada esporadicamente em projetos de DDS. Sua adoção se dá principalmente na fases iniciais do projeto com o objetivo da equipe conhecer os membros envolvidos no projeto. Ao longo do projeto, a sua adoção vai diminuindo. Os participantes descrevem a tecnologia de vídeo conferência como um ótimo recurso para facilitar a comunicação, embora não sendo muito adotado. De acordo com os participantes, equipes distribuídas de software preferem manter a comunicação de maneira impessoal, onde o foco é somente o áudio. Além disso, os participantes reportaram que muitos projetos não adotam a tecnologia de vídeo conferência, nem mesmo nas fases iniciais do projeto.

Os participantes também reportaram a adoção de viagens de trabalho entre sites para realizar a comunicação. Segundo os participantes, as viagens são realizadas durante as fases iniciais do projeto para transferir conhecimento entre sites ou iniciar um novo domínio de aplicação. Entretanto, ao longo dos anos essa estratégia de comunicação vem sendo menos adotada por organizações devido ao aumento de custos e os avanços da tecnologia.

4.2. Fatores que Afetam a Comunicação

Em projetos de DDS diferentes fatores podem afetar a comunicação. Tais fatores resultam em falhas de comunicação que consequentemente levam a crises no projeto. Entre esses fatores podemos citar o não entendimento de requisitos, a falta de consenso para resolver problemas técnicos, a falta de confiança entre os membros da equipe, etc. A Tabela 2 apresenta os principais fatores que afetam a comunicação segundo o relato dos participantes.

Tabela 2. Fatores que afetam a comunicação.

Id.	Fatores
1	Falta de comprometimento e engajamento da equipe
2	Excesso de e-mails trocados entre a equipe
3	Atrasos para receber *feedback* e atualizações de outros sites
4	Falta de conexão com a Internet durante viagens de trabalho
5	Falta de compreensão do idioma
6	Informação incompleta ou incorreta
7	Atualizações de infraestrutura durante o projeto

A falta de comprometimento e engajamento é vista pelos participantes como o principal fator que afeta a comunicação e que resulta em falhas do projeto. Um típico cenário onde os membros da equipe não estão comprometidos ou engajados constantemente apresenta atrasos para dar ou receber *feedback* e atualizações. A equipe está desmotivada e há uma excessiva troca de e-mails entre a equipe, onde percebe-se que não há progresso ou nenhuma conclusão é atingida.

De acordo com os participantes, a cultura dos brasileiros impõe que tudo seja documentado e que todas as pessoas envolvidas sejam copiadas nos e-mails. O uso do e-mail é visto como uma forma de documentar os problemas e as soluções dadas a estes. As reuniões por telefone ou ainda *conference calls* somente são utilizadas em casos emergenciais. Em alguns casos, os membros da equipe estão a poucos metros distantes um dos outros. Além disso, o uso de diferentes palavras para um mesmo significados é um exemplo de frequentes equívocos que acontecem na troca de e-mails.

Falhas na comunicação também ocorrem quando a informação trocada entre a equipe é incompleta ou incorreta. Os participantes reportam principalmente erros nas estimativas de prazos dadas pela equipe para completar uma tarefa e a falta de compreensão do idioma, uma vez que nem todos os membros da equipe são proficientes no idioma inglês. Em alguns casos, as falhas de comunicação resultam em dois ou três dias de atraso no projeto. Ainda, os participantes reportam que, embora haja motivação dos membros de equipes para aprender um novo idioma, equívocos de comunicação são frequentes. Ainda com relação ao idioma, os participantes relatam que um dos principais desafios para realizar a comunicação efetiva em projetos distribuídos é a descoberta da informação que está sendo solicitada ou transferência da informação. Devido a falta de proficiência no idioma, a informação precisa ser repetida inúmeras vezes para ser entendida pela equipe.

A comunicação na forma escrita, bastante frequente em projetos distribuídos, também é motivo de preocupações. Por exemplo em fóruns, as equipes possuem tempo limitado, o qual em alguns casos não é suficiente para repetir a informação ou tirar dúvidas. Assim, as equipes seguem o seu entendimento da informação que é fornecida. Além disso, as diferenças de fuso-horários aumentam o tempo para receber ou enviar uma resposta.

A conexão de Internet algumas vezes falha ou não está disponível durante viagens. Embora sendo possível observar que há uma maior disponibilidade de pontos de acesso a Internet em cafés e hotéis, a realização da comunicação baseada em rede sem fio pública ainda é difícil, devido a instabilidade da rede. Os participantes reportaram que a conexão ou não é confiável ou é constantemente interrompida.

Atualizações na infraestrutura durante o projeto é outro fator que causa falhas de comunicação. Em alguns casos, atualizações são realizadas sem um cronograma previamente definido em consequência de um pedido urgente, impactando no cronograma do projeto.

4.3. Estratégias para Mitigar Problemas de Comunicação

Os participantes também foram questionados sobre estratégias para mitigar as falhas de comunicação em projetos DDS. Considerando as informações dadas pelos participantes sobre as falhas de comunicação, foram elencadas 11 estratégias para mitigar os desafios de comunicação como mostra a Tabela 3.

Para mitigar a falta de engajamento e comprometimento da equipe, um dos participantes descreve duas abordagens, uma de baixo custo e outra de alto custo. Na primeira

Tabela 3. Estratégias de comunicação.

Id.	Nome da estratégia de comunicação
1	Desenvolvimento de *conference calls* semanais
2	Reuniões presenciais com os líderes da equipe
3	Regras para o uso de e-mail e telefone
4	Estabelecimento de um limite de tempo para responder um e-mail ou IM
5	Estímulo ao desenvolvimento relacionamentos profissionais e ciclos de trabalho
6	Revisão da documentação do projeto
8	Estabelecimento de reuniões documentadas
9	Gerenciamento de informações de membros críticos na equipe
10	Estabelecimento de uma hierarquia de comunicação
11	Desenvolvimento de competências no idioma adotado pela organização

abordagem é sugerido o emprego de reuniões por meio de *conference calls* semanais para unir a equipe e resolver problemas de engajamento. Com o contato semanal, os membros da equipe poderão se conhecer melhor, o que facilitará a transferência do conhecimento. Já na segunda abordagem é sugerido que alguns membros da equipe passem a trabalhar no mesmo ambiente físico por um certo período de tempo. Os líderes das equipes devem ser identificados e enviados para trabalhar em conjunto em um mesmo site. Recomenda-se que eles passem 2 ou 3 semanas trabalhando juntos e então retornem ao local de trabalho original. Entretanto, essa é uma abordagem que requer mudança temporária de local de trabalho, e impacta em custos com deslocamento e/ou hospedagem.

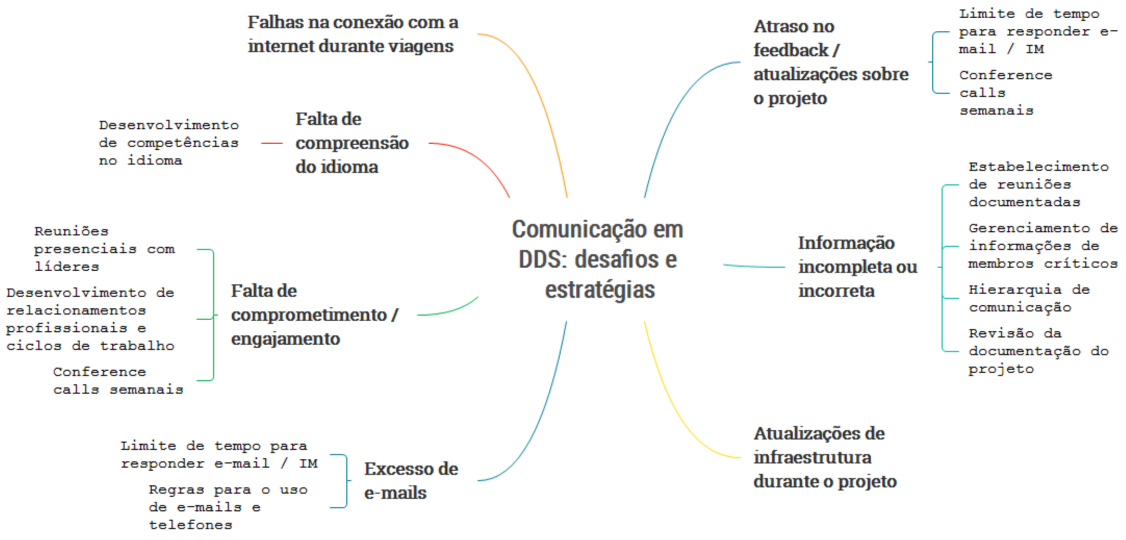
Em relação ao excesso de e-mails trocados entre os membros da equipe, os quais têm por objetivo achar uma solução para um problema, os participantes sugerem o desenvolvimento de reuniões semanais e o estabelecimento de regras para o uso do e-mail e do telefone. Os membros da equipe primeiro devem entender o que eles desejam comunicar e então escolher o melhor canal de comunicação. Algumas regras para melhorar a comunicação incluem o estabelecimento de um limite de tempo para responder um e-mail ou uma mensagem instantânea.

Muitos equívocos que ocorrem na comunicação são devidos à falta de confiança entre os membros da equipe. A construção de estreitas relações de confiança entre os membros e principalmente com os gerentes de projeto, resultam na troca de informações incompletas ou inconsistentes. Para mitigar esse problema, um dos participantes sugere a coleta de informações de um membro crítico da equipe e a revisão da documentação do projeto para mitigar as falhas. Além disso, é recomendado que as reuniões com a equipe sejam documentadas e que sejam feitas retrospectivas para identificar as lições aprendidas e melhorar o processo. Se uma decisão for tomada ou alguma mudança for feita, tal informação deve ser enviada por e-mail para todas as partes interessadas nessa informação.

Em projetos de DDS é comum que a equipe esteja dividida em diferentes fuso-horários, o que pode limitar a comunicação síncrona. O controle do número de membros da equipe em cada reunião e o repasse dessa informação para os seus gerentes pode aumentar o engajamento e o comprometimento da equipe. O estabelecimento de uma hierarquia de comunicação reduz a indisciplina e insubordinação da equipe.

A maioria das equipes que trabalham em projetos de DDS não são proficientes no idioma inglês. Para reduzir os problemas de comunicação relacionados ao idioma, os participantes recomendam que as organizações ofereçam workshops e cursos de inglês para qualificar as equipes.

Na Figura 1, as estratégias reportadas pelos participantes são associados aos fatores que afetam a comunicação. Para alguns fatores não foram identificadas estratégias de mitigação.



Comunicação em DDS: desafios e estratégias
Falhas na conexão com a internet durante viagens
Falta de compreensão do idioma
Desenvolvimento de competências no idioma
Falta de comprometimento / engajamento
Reuniões presenciais com líderes
Desenvolvimento de relacionamentos profissionais e ciclos de trabalho
Conference calls semanais
Excesso de e-mails
Limite de tempo para responder e-mail / IM
Regras para o uso de e-mails e telefones
Atraso no feedback / atualizações sobre o projeto
Limite de tempo para responder e-mail / IM
Conference calls semanais
Informação incompleta ou incorreta
Estabelecimento de reuniões documentadas
Gerenciamento de informações de membros críticos
Hierarquia de comunicação
Revisão da documentação do projeto
Atualizações de infraestrutura durante o projeto

Figura 1. Fatores que afetam a comunicação e estratégias de mitigação.

5. Discussão

A análise dos dados consistiu em aceitar as informações dadas pelos participantes baseada na experiência prática de trabalho de cada participante. Obstáculos como falta de comprometimento, falta de engajamento, atraso nos *feedbacks* e informações incorretas ou incompletas são exemplos de contratempos que impactam na qualidade da comunicação, e que são enfrentados em projetos de DDS ao redor do mundo. Especificamente no Brasil, problemas na conexão com a Internet em locais públicos merecem destaque, ao dificultar a comunicação em momentos em que o gerente de projeto ou outros membros precisam viajar para visitar os diferentes sites. Outro tópico que se destaca no cenário brasileiro é a falta de profissionais proficientes em inglês, que é o idioma mais adotado em projetos de DDS. Um aspecto que é inerente à cultura brasileira, refere-se ao uso de e-mails. Brasileiros preferem documentar todas a informações trocadas no decorrer do projeto para se resguardar de qualquer ação posterior, ao contrário de outras culturas que preferem a comunicação síncrona e pouco uso do e-mail.

Analisando os fatores apontados pelos entrevistados, observa-se a necessidade do planejamento da comunicação considerando a cultura brasileira. Um plano de comunicação bem elaborado deve contemplar estratégias como limite de tempo para responder e-mails, regras para uso de e-mails e telefones, modo esperado de documentação das reuniões, gerenciamento de expectativas de partes interessadas e hierarquia de comunicação.

A aplicação da boa prática "elaborar o plano de comunicações", descrita no [PMI 2013] poderia trazer benefícios para essas equipes.

6. Limitações

Esta seção discute algumas limitações deste estudo, as quais precisam ser destacadas. Foram entrevistados 10 gerentes de projeto, localizados em diferentes estados do Brasil. O número de participantes entrevistados pode ter influenciado nos resultados obtidos. Neste estudo não foi realizada a análise quantitativa dos dados devido ao tamanho da amostra, a qual planeja-se para um trabalho futuro com uma amostra maior de participantes. Também não é possível generalizar os resultados desse estudo, uma vez que os resultados refletem no cenário brasileiro de algumas organizações de DDS.

Os gerentes de projeto entrevistados neste estudo podem ter vivenciado diferentes cenários de comunicação em diferentes projetos e organizações. Isso ajuda a enriquecer as contribuições deste estudo. Entretanto, os resultados apresentados neste estudo limitam-se a expressar uma visão geral de ambientes, falhas e estratégias para mitigar lacunas de comunicação, sem especificar um cenário específico de comunicação.

Apesar destas limitações, espera-se que os resultados deste estudo forneçam informações que possam ajudar a compreender os principais aspectos de comunicação relacionados a ambientes, falhas e estratégias de mitigação em projetos de DDS.

7. Conclusões

Esse estudo investigou o ambiente de comunicação em projetos DDS na indústria brasileira de TI. Como resultado, foi identificado um conjunto de fatores que que afetam a comunicação em projetos de DDS e estratégias para reduzir problemas de comunicação. A partir desses resultados, observou-que que a comunidade brasileira de DDS ainda enfrenta vários problemas de comunicação, os quais mostram que o DDS ainda está em fase de aprimoramento e evolução dentro de organizações de TI no Brasil.

Com os resultados deste estudo ainda é possível observar que cenário brasileiro de DDS é carente de boas práticas de comunicação, o que consequentemente resulta em falhas e crises no projeto. Os fatores que afetam a comunicação refletem principalmente em aspectos culturais, de gerenciamento da informação e problemas de infraestrutura.

As estratégias de mitigação visam reduzir problemas de comunicação do cenário brasileiro de DDS, mas também podem ser aplicadas para outros contextos, os quais possuem um ambiente de comunicação similar ou parecido com o do Brasil.

Como trabalhos futuros, planeja-se dar continuidade a este estudo tento um maior aprofundamento dos fatores que resultam em falhas de comunicação e estratégias de mitigação. Novas entrevistas poderão ser conduzidas, como também a aplicação de *surveys* com gerentes de projetos, desenvolvedores e outros membros da equipe.

Referências

Anderson, A., McEwan, R., Bal, J., and Carletta, J. (2007). Virtual team meetings: An analysis of communication and context. *Computers in Human Behavior*.

Calefato, F., Lanubile, F., Conte, T., and Prikladnicki, R. (2012). Assessing the impact of real-time machine translation on requirements meetings: A replicated experiment. In

Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '12, pages 251–260, New York, NY, USA. ACM.

Colomo-Palacios, R., Casado-Lumbreras, C., Soto-Acosta, P., García-Peñalvo, F. J., and Tovar, E. (2014). Project managers in global software development teams: A study of the effects on productivity and performance. *Software Quality Journal*, 22(1):3–19.

Daim, T. U., Ha, A., Reutiman, S., Hughes, B., Pathak, U., Bynum, W., and Bhatla, A. (2012). Exploring the communication breakdown in global virtual teams. *International Journal of Project Management*.

Dingsoyr, T. and Smite, D. (2014). Managing knowledge in global software development projects. *IT Professional*, 16(1):22–29.

Herbsleb, J., Paulish, D., and Bass, M. (2005). Global software development at siemens: experience from nine projects. *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.*

Jimenez, M., Piattini, M., and Vizcaıno, A. (2009). Challenges and improvements in distributed software development: a systematic review. *Advances in Software Engineering Volume 2009, January 2009 Article No. 3.*

Mishra, D. and Mahanty, B. (2015). A study of software development project cost, schedule and quality by outsourcing to low cost destination. *Journal of Enterprise Information Management, Vol. 29 No. 3, 2016, pp. 454-478.*

Nguyen-Duc, A., Cruzes, D. S., and Conradi, R. (2015). The impact of global dispersion on coordination, team performance and software quality – a systematic literature review. *Information and Software Technology*, 57:277 – 294.

Paasivaara, M., Durasiewicz, S., and Lassenius, C. (2008). Using scrum in a globally distributed project: A case study. *Software Process Improvement and Practice*.

Paasivaara, M. and Lassenius, C. (2003). Collaboration practices in global inter-organizational software development projects. *Software Process Improvement and Practice*.

PMI (2013). *Um Guia do Conhecimento em Gerenciamento de Projetos (Guia PMBOK®). — Quinta edição*. Project Management Institute, Inc.

Prikladnicki, R. (2012). Propinquity in global software engineering: examining perceived distance in globally distributed project teams. *Journal of Software: Evolution and Process*.

Sievi-Korte, O., Systä, K., and Hjelsvold, R. (2015). Global vs. local - Experiences from a distributed software project course using agile methodologies. *Proceedings - Frontiers in Education Conference, FIE.*

Stawnicza, O. (2015). Distributed team cohesion – not an oxymoron . The impact of information and communications technologies on teamness in globally distributed IT projects. *International Journal of Information Systems and Project Management*.

Weimann, P., Pollock, M., Scott, E., and Brown, I. (2013). Enhancing team performance through tool use: How critical technology-related issues influence the performance of virtual project teams. *IEEE Transactions on Professional Communication*.

Ecossistema para o domínio educacional: modelo baseado em serviços Web

Welington Veiga, Fernanda Campos, Regina Braga, José Maria N. David, Victor Ströele

Universidade Federal de Juiz de Fora – Programa de Pós-graduação em Ciência da Computação – Núcleo de Pesquisa em Engenharia do Conhecimento
Juiz de Fora – MG – Brasil

welington.veiga@ice.ufjf.br, fernanda.campos@ufjf.edu.br, regina.braga@ufjf.edu.br, jose.david@ufjf.edu.br, victor.stroele@ice.ufj.br

Abstract. *This paper presents BROAD-ECOS approach, under Software Ecosystem perspective, and defines a platform to allow existing Virtual Learning Environments integrate external educational services, promoting the development, sharing and reuse of compatible educational services in an inter-organizational context. The platform evaluation was performed in a real use scenario, which goal was to demonstrate the technical feasibility of the proposal.*

Resumo: *Esse artigo apresenta o BROAD-ECOS, uma abordagem sob a perspectiva de Ecossistemas de e-Learning, que define uma plataforma que permite a integração de serviços educacionais externos e favorece o desenvolvimento, compartilhamento e reúso de serviços educacionais em um contexto interorganizacional. A avaliação da plataforma foi feita com um cenário de uso real, com o objetivo de demonstrar a viabilidade técnica da plataforma.*

1. Introdução

A produção de recursos e serviços educacionais é fundamental para a criação de ambientes de e-Learning que proporcionem uma experiência de ensino e aprendizagem rica. A aprendizagem informal, hoje facilitada pelos dispositivos móveis, e a aprendizagem ao longo da vida abrem novos possibilidades e desafios no uso desses ambientes. Por outro lado, uma das características do domínio educacional é o número de soluções específicas e fragmentadas nessas plataformas [Fragoso *et al*, 2014]. Um dos desafios nas organizações que proveem Ambientes Virtuais de Aprendizagem (AVAs), é oferecer recursos e serviços educacionais equivalentes aos já existentes e mais inovadores e que estes recursos atendam a esse nível de exigência e complexidade com os quais os usuários estão acostumados.

Esse cenário é interessante para a adoção da perspectiva de Ecossistemas de Software (ECOS) em ambientes de e-Learning. Manikas (2016) revisitou o conceito de ecossistema de software e o definiu como a interação entre software e atores em relação a uma infraestrutura tecnológica comum, que resulta em um conjunto de contribuições e influencia direta ou indiretamente o ecossistema. É uma forma de entender domínios em que diferentes organizações se relacionam por meio de software ou conceitos de software [Jansen e Cusumano, 2012].

Assim, a motivação do presente trabalho é aplicar, no domínio educacional e particularmente nos ambientes de e-Learning, a perspectiva de Ecossistemas de Software, permitindo que diferentes organizações contribuam com soluções e inovação para a construção de ambientes educacionais cada vez mais ricos em diversidade de conteúdo, serviços, experiências e capacidade de atender aos seus objetivos educacionais dentro do paradigma pedagógico adotado. Uma motivação adicional é avançar as pesquisas do projeto BROAD [Rezende *et al*, 2015], [Pereira *et al*, 2015] [Veiga *et al*, 2015], do Núcleo de Pesquisa em Engenharia do Conhecimento – NEnC.

A questão de pesquisa deste trabalho pode ser assim enunciada: como os ambientes de e-Learning, transformados em plataformas sob a perspectiva de Ecossistemas de Software, com uma infraestrutura para integração de ferramentas externas em um contexto interorganizacional, favorecem o compartilhamento e reúso de soluções e a utilização de recursos educacionais? O BROAD-ECOS é voltado para a criação de uma plataforma comum, disponibilizando ferramentas e documentação para que desenvolvedores, comunidades e organizações desenvolvam, compartilhem e reutilizem soluções, considerando as características específicas do domínio educacional com foco em e-Learning, e, sobretudo, as soluções já existentes neste domínio.

Esse artigo está assim organizado. Na seção 2 apresentam-se os fundamentos teóricos da pesquisa. Na seção 3 é apresentado o ecossistema BROAD-ECOS incluindo a avaliação de sua plataforma. Na seção 4 relatam-se os trabalhos relacionados. Na seção 5 fazem-se as considerações finais.

2. Contexto da Pesquisa

A fragmentação das soluções no domínio educacional, a crescente complexidade dos softwares, o surgimento de diferentes formas de aprendizagem, aliado ao grau de exigência dos usuários expostos a aplicações e serviços de software de alta complexidade, além da falta de um padrão estabelecido para incorporar soluções externas aos ambientes de e-Learning, são desafios que as organizações mantenedoras de AVAs têm dificuldade para atender sozinhas.

Neste contexto, serviços Web apoiam a interoperabilidade entre aplicações, que podem ser executadas em diferentes plataformas, *frameworks* e tecnologia. No domínio educacional, é discutido o uso de serviços como forma de adicionar funcionalidades, integrar diferentes ambientes de e-Learning [Boeringer, 2014] e oferecer experiências como a utilização de serviços Educacionais a partir de dispositivos móveis [Alzaza, 2011]. O uso de Serviços Educacionais em AVA, além de facilitar o reúso entre plataformas heterogêneas [D'Mello *et al*, 2013], pode melhorar a interoperabilidade, flexibilidade, reusabilidade e compatibilidade entre plataformas.

Por outro lado, a conectividade e interdependência entre empresas é um assunto de crescente interesse [Barbosa et al, 2013]. As empresas não funcionam mais como unidades independentes, o que as tornam dependentes de componentes e infraestrutura fornecidos por terceiros. Nesse contexto, pesquisadores utilizam uma nova perspectiva para analisar a indústria de software, os Ecossistemas de Software (ECOS). Estes, representam um ponto de vista inspirado em ecossistemas naturais e de negócios, através dos quais considera-se não só o software em si, mas suas dependências de componentes/infraestrutura de terceiros, seus usuários e suas interações.

Neste trabalho, será utilizada a definição de ECOS proposta por Manikas [2016], expandida pelos aspectos identificados em Santos [2016], que oferecem um nível de compreensão importante dos diferentes fatores de um ECOS, capturando as interações, motivações, relações, papéis e conceitos, além dos aspectos técnicos da plataforma.

3. BROAD-ECOS

Um dos objetivos estabelecidos para essa proposta é viabilizar a sua adoção em AVAs existentes, incorporando a infraestrutura necessária para que estes ambientes façam parte de uma plataforma de ecossistema. A proposta da plataforma BROAD-ECOS pode ser dividida em duas partes: a primeira é a proposta de compreensão dos ambientes de e-Learning sob a perspectiva de ECOS, identificando sua composição, atores e formas de interação em um contexto inter-organizacional. A segunda é a definição de uma plataforma que englobe os AVAs como principais *players*, permitindo que Serviços Educacionais externos sejam integrados.

Nesta perspectiva, um Ecossistema de e-Learning pode ser definido como um ECOS do domínio educacional no contexto de e-Learning, onde os componentes abióticos (como AVA, Serviços Educacionais, Objetos de Aprendizagem (OAs), mídias) e bióticos (como estudantes, facilitadores, especialistas, suporte, apoio, conteudistas, desenvolvedores) interagem e formam comunidades em um contexto inter-organizacional voltados ao processo de ensino e aprendizagem, delimitados por fronteiras pedagógicas, sociais, econômicas e culturais. A Figura 1 apresenta uma visão geral do BROAD-ECOS, ressaltando os atores e serviços que interagem com a sua plataforma.

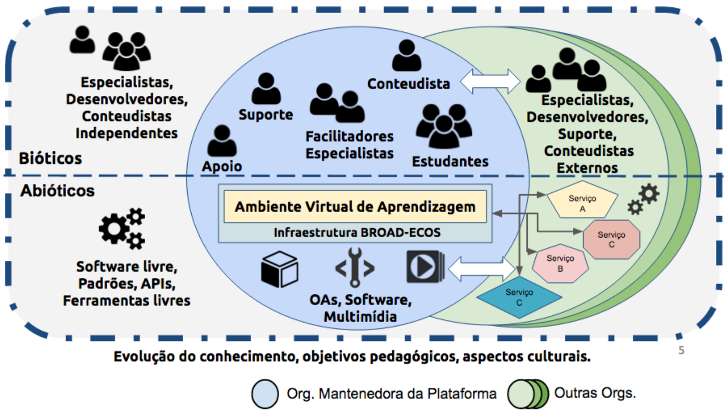

Especialistas, Desenvolvedores, Conteudistas Independentes
Bióticos
Abióticos
Software livre, Padrões, APIs, Ferramentas livres
Suporte
Apoio
Facilitadores Especialistas
Conteudista
Estudantes
Especialistas, Desenvolvedores, Suporte, Conteudistas Externos
Ambiente Virtual de Aprendizagem
Infraestrutura BROAD-ECOS
OAs, Software, Multimídia
Serviço A
Serviço C
Serviço B
Serviço C
Evolução do conhecimento, objetivos pedagógicos, aspectos culturais.
5
Org. Mantenedora da Plataforma
Outras Orgs.

Figura 1 – Visão geral de Ecossistemas de e-Learning sob a abordagem BROAD-ECOS.

No contexto do ecossistema BROAD-ECOS, os principais componentes foram propostos a partir de uma pesquisa exploratória, realizada por meio de um questionário composto por perguntas fechadas. A partir dos resultados da pesquisa, experiência dos autores com ambientes educacionais e com base em uma revisão *quasi* sistemática, os componentes foram reavaliados e a proposta do ecossistema BROAD-ECOS foi delimitada, conforme Figura 1. Os principais componentes são detalhados a seguir.

Os fatores bióticos no ecossistema de e-Learning não se restringem a indivíduos, incluindo também comunidades de indivíduos com os mesmos interesses ou papeis e organizações compostas por diversos indivíduos que compartilham sua missão e objetivos. É importante destacar que em um ambiente de e-Learning, um fator biótico pode atuar em diferentes papéis de acordo com o contexto, e até mesmo acumular funções. Desse modo, os papéis devem ser vistos como uma função no ambiente de e-Learning, e não como um ator específico desempenhando determinado papel.

Os fatores abióticos são compostos por todos os recursos de software no processo de ensino e aprendizagem, e abrangem desde ferramentas instaladas nos computadores dos atores, a serviços disponíveis na nuvem e integrados à plataforma. Entre os fatores abióticos, a plataforma merece destaque, sendo composta no BROAD-ECOS pela infraestrutura integrada ao AVA. Seu papel é particular, primeiro por ser a responsável por informações de cursos, turmas e participantes, e segundo pela orquestração dos serviços externos, definindo permissões, recursos disponíveis e grau de integração dos mesmos no ecossistema.

OAs e multimídia são componentes importantes por poderem ser desenvolvidos por diferentes organizações e compartilhados entre diferentes AVAs. Os serviços educacionais podem ser desenvolvidos por diferentes organizações, e possuir diferentes níveis de integração, com diversas possibilidades de aplicação e objetivo educacional. Os padrões, recomendações técnicas, especificações e APIs são convenções que permitem que diferentes organizações possam compartilhar recursos e estabelecer relações de cooperação e competição essenciais para a saúde do ecossistema.

3.1 Plataforma

A plataforma proposta visa adicionar aos AVAs existentes os recursos necessários para que estes sejam ambientes abertos com suporte à integração de serviços externos, em um contexto inter-organizacional, a partir do qual o ECOS proposto se sustenta.

Inicialmente, foram definidas as seguintes características que devem ser adicionadas aos AVAs através da plataforma: i. **Integração de serviços externos** (adicionar ao ecossistema serviços com diferentes funcionalidades e objetivos sem que seja necessário alterar a plataforma, facilitando a integração de serviços de outras organizações); ii. **Estabelecimento de um modelo comum para o domínio educacional** (mesmo que o AVA e os serviços possuam abstrações diferentes para as entidades, a comunicação deve ser feita a partir de uma abstração comum do domínio); iii. **Controle de autorização** (oferecer uma forma de autorização para acessos de serviços a recursos oferecidos na plataforma, de baixa granularidade); iv. **Uso de metadados educacionais** (facilitar a pesquisa e seleção de serviços educacionais adequados); v. **Registro e recuperação de experiências de aprendizagem** (coleta e acompanhamento das experiências educacionais que não se resumam a notas de avaliações e respostas de questionário) e vi. **Suporte à existência de um canal de distribuição** (repositório onde os facilitadores e especialistas possam buscar serviços). Existem ainda características que consistem em requisitos não funcionais: extensibilidade (permite que necessidades específicas sejam atendidas mantendo a compatibilidade com a abordagem BROAD-ECOS) e poucas restrições na criação de serviços compatíveis (elemento preponderante para os ecossistemas).

A plataforma utiliza uma arquitetura orientada a serviços Web e toda comunicação entre os componentes é feita através de serviços RESTful com dados em formato JSON [Marinos *et al*, 2011], além do suporte de bibliotecas, *frameworks* e ferramentas, independentes de linguagens de programação e SO (Figura 2). O conjunto de serviços Web disponíveis para a comunicação entre os serviços Educacionais e a plataforma formam a BROAD-ECOS-API, a qual define os recursos, a estrutura das requisições e as possíveis respostas relacionadas às entidades externas que interagem com a plataforma, e apoia a sua extensibilidade. É um modelo de permissões com fina granularidade, garantindo flexibilidade no nível de integração entre cada serviço e a plataforma. A segurança dos dados é dada pelo BROAD-ECOS-Auth, uma adaptação do protocolo OAuth 2.0 [OAUTH, 2015].

Além de suportar operações sobre o domínio educacional com entidades como participantes, turmas e cursos, a BROAD-ECOS-API suporta ainda todos os recursos da xAPI *Statements*, que permite o registro e recuperação de experiências educacionais em repositórios chamados de LRS (*Learning Record Store*). Uma parte da BROAD-ECOS-API ainda é responsável por permitir o registro e a recuperação de metadados de serviços educacionais em outro componente da plataforma, o BROAD-ECOS-DC. Este componente é um canal de distribuição de serviços, que não contém os serviços propriamente ditos, mas seus metadados compostos pelo padrão IEEE LOM[1] enriquecido com características específicas do BROAD-ECOS, como as permissões exigidas e a versão da BROAD-ECOS-API suportada, por exemplo.

A conexão entre a plataforma e o AVA é feita através do *Adapter*, uma camada de software na forma de *plugin*, extensão ou alteração no próprio AVA. É responsável por traduzir as chamadas à BROAD-ECOS-API, que são independentes do AVA, para as peculiaridades e características de cada um deles. Para a criação de Serviços Educacionais compatíveis com a plataforma, basta manter a compatibilidade com a BROAD-ECOS-API que exige que apenas três recursos web sejam suportados pelo serviço (para atender ao protocolo de segurança e disponibilizar metadados) e utilizar apenas os recursos do lado da plataforma que desejar, ignorando os demais.

Os recursos oferecidos pela plataforma pertencem a cinco grupos: recursos de acesso/atualização de dados do domínio educacional; recursos de consulta e envio de experiências de aprendizagem, compatíveis com a xAPI *Statement*; recursos de autenticação e autorização, de acordo com o protocolo BROAD-ECOS-Auth; recursos para adicionar e recuperar serviços do canal de distribuição BROAD-ECOS-DC; um pequeno conjunto de Serviços Web que os Serviços Educacionais devem suportar, para informar metadados e atender ao BROAD-ECOS-Auth, por exemplo; e serviços suportados a partir de extensões, que podem ser negociadas entre a plataforma e os serviços. No Grupo I, estão os recursos relacionados ao domínio de e-Learning. Para a comunicação entre os Serviços Educacionais e o AVA, fez-se necessária a definição de um modelo de domínio educacional compartilhado, cuja formalização foi definida numa ontologia, descrita utilizando a linguagem OWL (Figura 3).

[1]https://standards.ieee.org/findstds/standard/1484.12.1-2002.html

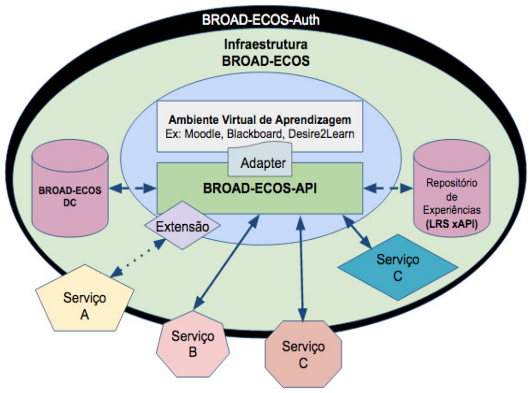

BROAD-ECOS-Auth
Infraestrutura BROAD-ECOS
Ambiente Virtual de Aprendizagem
Ex: Moodle, Blackboard, Desire2Learn
Adapter
BROAD-ECOS-API
BROAD-ECOS DC
Repositório de Experiências (LRS xAPI)
Extensão
Serviço A
Serviço B
Serviço C
Serviço C

Figura 2 – Componentes da Plataforma BROAD-ECOS.

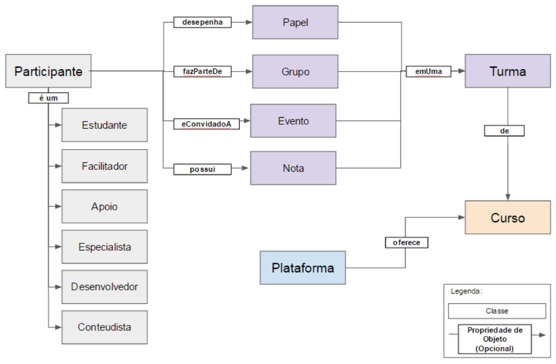


Figura 3 - Ontologia de domínio dos principais participantes.

4. Utilização da plataforma BROAD-ECOS no Moodle

Para demonstrar a viabilidade da plataforma em relação à integração de serviços educacionais externos ao AVA do ponto de vista de atores do ecossistema foi utilizado um cenário, com as seguintes fases: (I) definição; (II) formulação do objetivo; (III) planejamento; (IV) execução e observação das evidências; e (V) a apresentação das evidências observadas. Para a avaliação da plataforma em uso foi escolhido o AVA Moodle, de código livre e com tecnologias abertas e suporte a *plugins* de terceiros.

Para atender ao requisito de baixo acoplamento foi utilizado o padrão de projeto estrutural *Adapter*. Na BROAD-ECOS, o *Adapter* é a camada de software que estará integrada ao AVA, traduzindo as chamadas à BROAD-ECOS-API para suas estruturas internas, e deve permitir o reuso de tudo o que for comum entre AVAS diferentes.

Foi desenvolvido um *plugin* para a BROAD-ECOS chamado *mod-broad-ecos*, que inclui uma implementação da infraestrutura BROAD-ECOS na linguagem de programação PHP e um *Adapter* que abstrai as chamadas específicas aos recursos do Moodle e disponibiliza interfaces de usuário dentro do AVA para adição e acesso de serviços como atividades nos cursos oferecidos através da plataforma. O *plugin* depende da existência de um repositório de experiências externo. Para essa versão, foi utilizado o *Learning Locker*, uma implementação livre de LRS para a xAPI (Figura 4).

Como existem muitos componentes e configurações necessárias para conectar os serviços, foi utilizada a estratégia de *containers* [Boettiger, 2015]. A implementação de *containers* utilizada foi o Docker[2], em conjunto com a ferramenta docker-compose[3], utilizada para automatizar o *setup* do ambiente com múltiplos *containers*.

As funcionalidades disponíveis no *plugin* do BROAD-ECOS para o Moodle são: (I) adição/edição de um serviço compatível com a BROAD-ECOS em determinado curso; (II) definição/edição do título e descrição do serviço no contexto em que ele é disponibilizado; (III) seleção/edição dos escopos aos quais o serviço deve possuir acesso no contexto em que foi adicionado; (IV) possibilidade dos estudantes acessarem o

[2] https://www.docker.com/
[3] https://docs.docker.com/compose/

serviço diretamente no Moodle; e (V) possibilidade de acessar o serviço externamente, mantendo-o em uma janela.

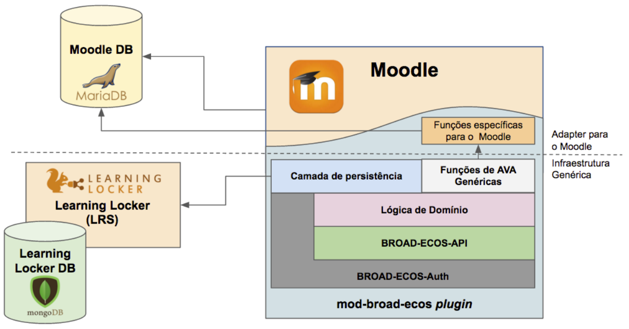
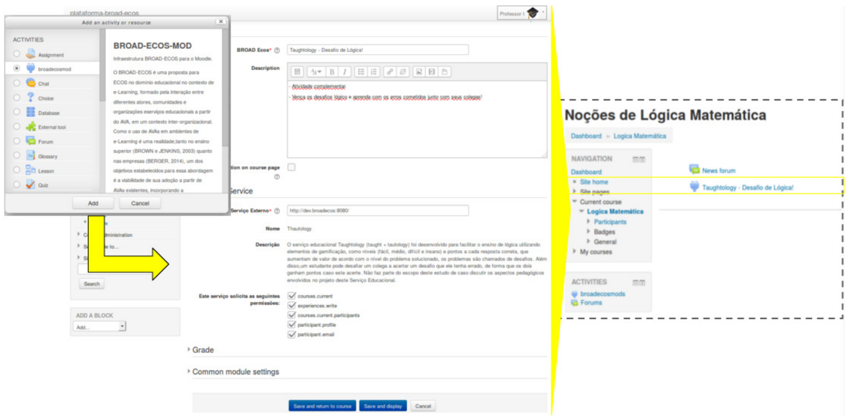


Figura 4 - Arquitetura do *plugin* mod-broad-ecos.

As etapas planejadas para a execução do cenário de uso foram: (Etapa I) Adição do AVA Moodle a plataforma BROAD-ECOS por um Apoio; (Etapa II) Inclusão de um serviço em um curso da plataforma por um Facilitador; e (Etapa III) Acesso a um serviço por um Estudante. Para ilustrar a Etapa II, o usuário Professor I acessa o Moodle e navega até a sua página de cursos, selecionando um de seus cursos, nesse cenário Introdução à Lógica. Na página de edição do curso, adiciona uma atividade e na tela seguinte seleciona o tipo de atividade “mod-broad-ecos”, e é redirecionado para a página de configuração da atividade criada (Figura 5). Na Etapa III, o Estudante I acessa o ambiente e navega até o curso Introdução à Lógica.


BROAD-ECOS-MOD

Infraestrutura BROAD-ECOS para o Moodle.

O BROAD-ECOS é uma proposta para ECOS no domínio educacional no contexto de e-Learning, formado pela interação entre diferentes atores, comunidades e organizações eserviços educacionais a partir do AVA, em um contexto inter-organizacional. Como o uso de AVAs em ambientes de e-Learning é uma realidade,tanto no ensino superior (BROWN e JENKINS, 2003) quanto nas empresas (BERGER, 2014), um dos objetivos estabelecidos para essa abordagem é a viabilidade de sua adoção a partir de AVAs existentes, incorporando a

BROAD Ecos*: Taughtology - Desafio de Lógica!

Description: - Atividade complementar
- Vença os desafios lógico e aprenda com os erros cometidos junto com seus colegas!

Serviço Externo*: http://dev.broadecos:8080/

Nome: Thautology

Descrição: O serviço educacional Taughtology (taught + tautology) foi desenvolvido para facilitar o ensino de lógica utilizando elementos de gamificação, como níveis (fácil, médio, difícil e insano) e pontos a cada resposta correta, que aumentam de valor de acordo com o nível do problema solucionado, os problemas são chamados de desafios. Além disso,um estudante pode desafiar um colega a acertar um desafio que ele tenha errado, de forma que os dois ganham pontos caso este acerte. Não faz parte do escopo deste estudo de caso discutir os aspectos pedagógicos envolvidos no projeto deste Serviço Educacional.

Este serviço solicita as seguintes permissões:
- ✓ courses.current
- ✓ experiences.write
- ✓ courses.current.participants
- ✓ participant.profile
- ✓ participant.email

Noções de Lógica Matemática

Dashboard ▸ Logica Matemática

Figura 5 - Adição de uma atividade broad-mod-ecos no curso.

4.2 Evidências Observadas

O cenário de uso apresenta indícios da viabilidade da proposta, dos conceitos e das tecnologias envolvidas na integração de serviços educacionais ao AVA Moodle, cobrindo um ciclo de conexão do AVA à plataforma, utilizando um *plugin*, a configuração de um curso por um Facilitador e o acesso por um Estudante. Esse cenário de uso apresentou os principais componentes da plataforma, o uso da BROAD-ECOS-API em conjunto com o BROAD-ECOS-Auth para garantir a comunicação entre a plataforma e o serviço, o uso de metadados para identificação do conteúdo do serviço educacional, o uso da xAPI para armazenar experiências de aprendizagem em um LRS e a abstração de características específicas do AVA através de um *Adapter*. Também foram utilizados os recursos para desenvolvedores de aplicações compatíveis com a BROAD-ECOS, como a biblioteca BroadEcosApi e a documentação. Durante sua execução, foi possível observar que, do ponto de vista de atores não técnicos como Apoio, Professor e Estudante o uso da BROAD-ECOS para a integração e acesso ao serviço externo não exige configurações técnicas.

A quantidade de atores e de serviços educacionais externos disponíveis no momento da realização da avaliação podem ter influenciado a análise dos resultados. Além disso, cabe ressaltar que a complexidade dos diferentes níveis de integração de serviços à plataforma pode influenciar na utilização dos mesmos pelos atores.

5. Trabalhos Relacionados

O Jampots [Dong *et al*, 2009] é uma implementação de Ecossistema de e-Learning baseado em *mashups*, combinação de informações de várias fontes em diferentes formatos oferecidos através de APIs de terceiros gerando novas funcionalidades, e constitui-se de uma plataforma para o projeto colaborativo, disponibilização, compartilhamento, gerenciamento e recriação de conteúdo educacional para usuários finais. O Dippler [Laanpere *et al*, 2013] é a plataforma de um ecossistema que não se limita à estrutura, arquitetura e implementação, mas considera uma perspectiva mais ampla, sócio tecnológica, envolvendo aspectos políticos, econômicos, acadêmicos e tecnológicos. A Experience API (xAPI), é uma especificação que permite a coleta de dados sobre uma ampla gama de experiências de aprendizagem, online e off-line. Permite que diferentes sistemas compartilhem informações educacionais utilizando um vocabulário compartilhado. É aberta, seu desenvolvimento é dirigido pela comunidade, e permite livre implementação [ADL 2016]. Devido a definição de uma interface e um formato de armazenamento de dados comuns, a xAPI estabelece uma especificação que suporta a definição de um Ecossistema de e-Learning [Hruska *et al*, 2015].

Sob a perspectiva de ECOS, os trabalhos apresentados não cobrem aspectos como a possibilidade de contribuição externa à plataforma, suporte de ferramentas e bibliotecas que favoreçam a construção de serviços compatíveis. Além disso, atendem parcialmente aspectos como o uso de ativos compartilhados e a possibilidade de incorporar recursos de terceiros à plataforma. Em relação às características de integração de serviços a ambientes de e-Learning, a plataforma BROAD-ECOS reforça o uso de padrões e recomendações do domínio de e-Learning e a disponibilização de documentação para a construção de serviços compatíveis, além de suportar recursos como o suporte a visões diferentes baseadas em papéis na utilização dos serviços, atividades e interação em grupo e o uso de adaptadores para a integração de serviços existentes. A previsão de colaboração externa à plataforma, o suporte à variabilidade, a possibilidade de incorporação de soluções de terceiros e a existência de um canal de distribuição de serviços educacionais aberto a organizações externas também estão presentes na BROAD-ECOS e não são mencionadas em trabalhos anteriores.

6. Considerações Finais

Este artigo apresentou uma plataforma extensível de ECOS no domínio de e-learning, o BROAD-ECOS, a partir da interação entre diferentes atores, comunidades, organizações e serviços educacionais numa plataforma tecnológica comum, com o objetivo de apoiar o desenvolvimento, compartilhamento e reuso de serviços educacionais em um contexto inter-organizacional. Este trabalho faz parte ainda do projeto BROAD, que engloba pesquisas relacionadas à investigação e adoção de tecnologias em projetos educacionais.

O suporte a diferentes níveis de integração entre serviços externos e a plataforma pode permitir o uso da BROAD-ECOS tanto para integrar jogos e atividades simples quanto módulos de gestão acadêmica completos, e ampliar as possibilidades de serviços educacionais compartilhados entre organizações. A ênfase em manter uma plataforma com poucas restrições para a criação de serviços pode facilitar o interesse de terceiros a desenvolver soluções compatíveis ou adaptar soluções existentes para a BROAD-ECOS. A diversidade é uma das características do domínio educacional, dessa forma, a definição de um mecanismo para extensão dos recursos previamente definidos é importante para que a proposta seja considerada mesmo em cenários em que existam necessidades específicas. Há indícios de que é possível propor um ecossistema mesmo em um domínio tão heterogêneo como o educacional, em contextos, objetivos educacionais, abordagens pedagógicas e necessidades organizacionais específicas.

Como trabalhos futuros, cabe ressaltar a disponibilização de um canal de distribuição, como uma loja de serviços educacionais através da qual os professores possam encontrar e adicionar serviços educacionais. Além disso, é importante a implementação de todos os recursos da BROAD-ECOS-API descritos e o aperfeiçoamento dos artefatos e da documentação necessária para permitir novas pesquisas e integração e uso por terceiros.

Agradecimentos: à FAPEMIG, CNPq, UFJF e Capes pelo apoio financeiro ao projeto.

Referências

ADL, Advanced Distributed Learning. Experience API (xAPI). Disponível em http://www.adlnet.gov/tla/experience-api Acessado em 12 de outubro de 2015.

Alzaza e Yaakub. (2011) "Students' awareness and requirements of mobile learning services in the higher education environment". American Journal of Economics and Business Administration. p. 95.

Barbosa, O. Santos, R.; Alves, C.; Werner, C. e Jansen, S. (2013). "A systematic mapping study on software ecosystems from a three-dimensional perspective". In: *Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry*. Edward Elgar Publishing, 2013.

Boehringer, D.; Bernlohr, H. (2014) CampusConnect: An open-source initiative to connect Learning Management Systems, Global Engineering Education Conference (EDUCON), 2014 IEEE, 2014.

D'Mello, D.; Crasta, M.; Gomes, R.; Abhishiktha, V.; D'Souza, S. (2013) "A Broker Based E-Learning Service Integration and Execution Architecture for the Complex Requirements on Learning Resources". Advanced Computing, Networking and Security (ADCONS), 2nd International Conference on. p. 35 - 40. IEEE. 2013.

Dong, B., Zheng, Q., Yang, J., Li, H. e Qiao, M. (2009) “An E-Learning Ecosystem Based on Cloud Computing Infrastructure”, Advanced Learning Technologies, 2009. ICALT 2009. Ninth IEEE Inter. Conf. on.pp.125, 127.

Fragoso, O. G., Santaolaya, R., Munoz, S. J., Valenzuela, B. D., e Rojas, J. C. (2014) “Integration of learning Web services into learning management systems”. In Central America and Panama Convention (CONCAPAN XXXIV), IEEE (pp. 1-6). 2014.

Hruska, M.; Medford., A; Murphy, J. (2015). Learning Ecosystems Using the Generalized Intelligent Framework for Tutoring (GIFT) and the Experience API (xAPI). AIED 2015 Workshop Proceedings, Vol. 6. 2015.

Jansen, S.; Cusumano, M. (2012) “Defining software ecosystems: A survey of software platforms and business network governance”. In: CEUR Workshop Proceedings, v. 879, p. 41–58. 2012.

Laanpere, M., Põldoja, H., & Normak, P. Designing Dippler—a Next-Generation TEL System. (2013) In Open and Social Technologies for Networked Learning (pp. 91-100). Springer Berlin Heidelberg. 2013.

Manikas, K (2016), “Revisiting Software Ecosystems Research: A Longitudinal Literature Study”. Journal of Systems and Software, 117, 84-103.

Marinos, A.; Moschoyiannis, S.; KRAUSE, P. (2011) “Towards a restful infrastructure for digital ecosystems”. International Journal of Electronic Business, v. 9, n. 5-6, p. 484-498, 2011.

OAUTH, OAuth 2. Disponível em<http://oauth.net/2/>.Acessoem 28 de Novembro de 2015.

Pereira, C. K.; Campos, F. C. A.; Stroele, V.; Braga, R. M.; David, J. M. N.; Almeida, R. (2015) Extração de Características de Perfil e de Contexto em Redes Sociais para Recomendação de Recursos Educacionais. In: Revista Brasileira de Informática na Educação (RBIE), v. 23, p. 25-39, 2015.

Rezende, P. A. A., Pereira, C. K., Campos, F., David, J. M. N., & Braga, R. (2015). "PERSONNA: proposta de ontologia de contexto e perfil de alunos para recomendação de objetos de aprendizagem". Revista Brasileira de Informática na Educação, 23(01), 70.

Santos, R (2016) “Managing and Monitoring Software Ecosystem to Support Demand and Solution Analysis”. PHD Thesis COPPE/UFRJ, disponível em http://reuse.cos.ufrj.br/site/pt/index.php?option=com_content&task=view&id=43&Itemid (acesso em maio de 2016).

Veiga, W.; Campos, F.; Braga, R.; David, J. M. (2015) “LUDOS: uma Infraestrutura para Gamificação em Ecossistemas de E-learning”. Anais do XXVI Simpósio Brasileiro de Informática na Educação (SBIE 2015). p. 459-469.

An Infrastructure to Support Socialization, Monitoring and Analysis of Software Ecosystems

Thaiana Lima, Gabriel Barbosa, Rodrigo Santos, Cláudia Werner

PESC/COPPE/UFRJ – System Engineering and Computer Science Department
Federal University of Rio de Janeiro – 21945-970 – Rio de Janeiro, RJ, Brazil

{thaiana, gabrielsb, rps, werner}@cos.ufrj.br

Abstract. *The software ecosystems context brings additional complexity to the organizations' activities since IT management decisions can strengthen (or weaken) relationships in the software supply network. Several acquirers lack common, structured documentation to allow visualizing and analyzing impacts of demands and solutions in their asset bases over time. We researched how the ecosystem perspective affects IT management activities, more specifically demand and solution analysis, aiming at helping IT managers and architects to make acquisition decisions. This paper presents an infrastructure to support socialization, monitoring and analysis of software ecosystems named Brechó-SocialSECO. A feasibility study was executed to evaluate our proposal.*

1. Introduction

According to Seichter et al. (2010), the artifacts shared among different stakeholders are the means for making interactions and communication concrete throughout the software development, and they create a common environment based on a technological platform. Such environment composed by actors, artifacts, and their relationships has been known as a Software Ecosystem (SECO). There is a network that represents those interactions, neither totally social (among actors) nor technical (among artifacts), i.e., the interactions involve actors manipulating artifacts (Lima et al., 2015).

Software organizations can benefit from information about the SECO elements, either a supplier or an acquirer. In this context, a critical acquirer's process is software acquisition, because it requires information about dependencies, technologies, suppliers, licensing, and support. Based on such elements, an acquirer should be able to choose which demands to prioritize and which solutions to incorporate into the organization's asset base (Finkelstein, 2014). However, several acquirers lack common documentation that is structured to allow visualizing and analyzing impacts of demands and solutions in their asset bases over time. So, the community participation (SECO *socialization*) and the asset base management (SECO *analysis*) are harmed so that obstacles are created for SECO monitoring. As such, an automated aid for managing and monitoring SECO elements becomes important (Santos, 2016).

We researched how the SECO perspective affects IT management activities, more specifically demand and solution analysis, aiming at helping IT managers and architects to make acquisition decisions. This paper presents an infrastructure to support socialization, monitoring and analysis of SECO that was evaluated through a feasibility study. Section 2 presents the background and related work; Section 3 describes our proposal; Section 4 discusses the feasibility study; and Section 5 concludes the paper.

2. Background

In the globalized software industry, there is a network composed by the SECO's elements and their relationships, including the involved organizations and community's demands. As such, the SECO context brings additional complexity to the organizations' activities since IT management decisions can strengthen (or weaken) relationships in the software supply network. For the traditional IT management, requirement specifications and available budget serve as key criteria to help IT managers and architects to analyze demands, lacking a structured software asset base to explore other indicators, especially the 'hidden effects' of their long-term decisions. In this intertwined network, acquirers operate relationships with/between suppliers and need to manage and monitor SECO elements in order to make decisions and take business advantages (Gartner, 2007).

The SECO context comes up with the importance of combining market information (inter-organizational) with those obtained from the software asset base (intra-organizational) to help IT managers and architects in daily activities (Finkelstein, 2014). To cope with this challenge, we have been investigating SECO management and extended a component library named Brechó (Werner et al., 2009). Brechó is a web information system for storage, publishing, search and retrieval, download, purchase and negotiation of software artifacts. Brechó is a technical repository that can serve as a software asset base, supporting licensing and SECO analysis. The lack of social resources to encourage stakeholders' interactions and communication had been pointed out as the Brechó's weakness until we conducted this research (Lima et al., 2015).

In the SECO literature, we can find few related work on the socialization, monitoring and/or analysis of SECOs. Peréz et al. (2012) present a tool named SECONDA that aims to aid SECO analysis and evolution according to project metrics and based on SECO behaviors and characteristics. However, this tool has some limitations, such as the lack of support for social relationships and for trend analysis from the collected information. The Brazilian Public Software (BPS) Portal is a web platform that offers a plethora of social interaction mechanisms, such as forums, communities, and a repository. BPS holds information on the actors' relationships, but there is no support for SECO monitoring and visualization. Goeminne et al. (2010) propose a framework for capturing SECO information based on extensions. By using data mining, SECO's graphics and a lifecycle chart are generated, but it is a generic framework that requires effort to program the information one needs to mine from it. All those mentioned gaps are part of the solution described in the proposed infrastructure.

3. Brechó-SocialSECO

Based on a survey conducted to investigate socio-technical resources for SECOs (Lima et al., 2015), we identified some social mechanisms as solutions for the problems of providing software artifact information and helping stakeholders to communicate: *artifact/team forum*, *team creation*, *actor/artifact profile*, *trend topics*, *suggestions and recommendations*, and *demand and solution analysis*. Such mechanisms can help IT managers and architects to be aware of SECO trends, and discussions would be of great value for the identification of new demands over time. In order to treat socialization in a SECO platform and aid an acquirer to satisfy objectives, stimulate collaboration and identify demands, Brechó-SocialSECO infrastructure was developed. Its objective is to help an acquirer to understand the community surrounding the SECO platform as well as map relationships, manage demands and visualize the evolution of social interaction.

It is essential to record how demands and candidate solutions, also labeled as *software assets*, are added, deleted, and maintained. Considering the lack of research that combines socio-technical network with data obtained from the acquirer's asset base, Brechó-SocialSECO was built upon SECO socialization and analysis mechanisms to support IT management activities, as shown in Figure 1. Social mechanisms are used as inputs to capture information from actors, artifacts and interactions. Such information is processed to generate the SECO network graph. Based on this graph, measures can be extracted so that IT managers and architects can make decisions based on different configurations of the SECO's network (SECO analysis). The infrastructure was implemented as an extension of Brechó (Section 2). All the mechanisms for the SECO socialization, monitoring and analysis are joined in the "My Network" panel (Figure 2).

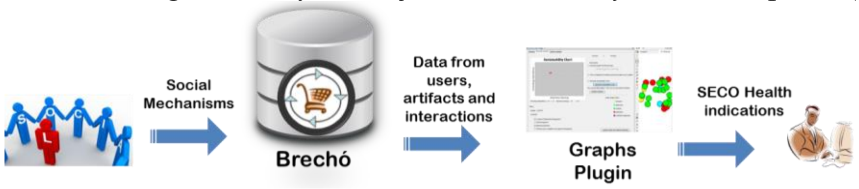
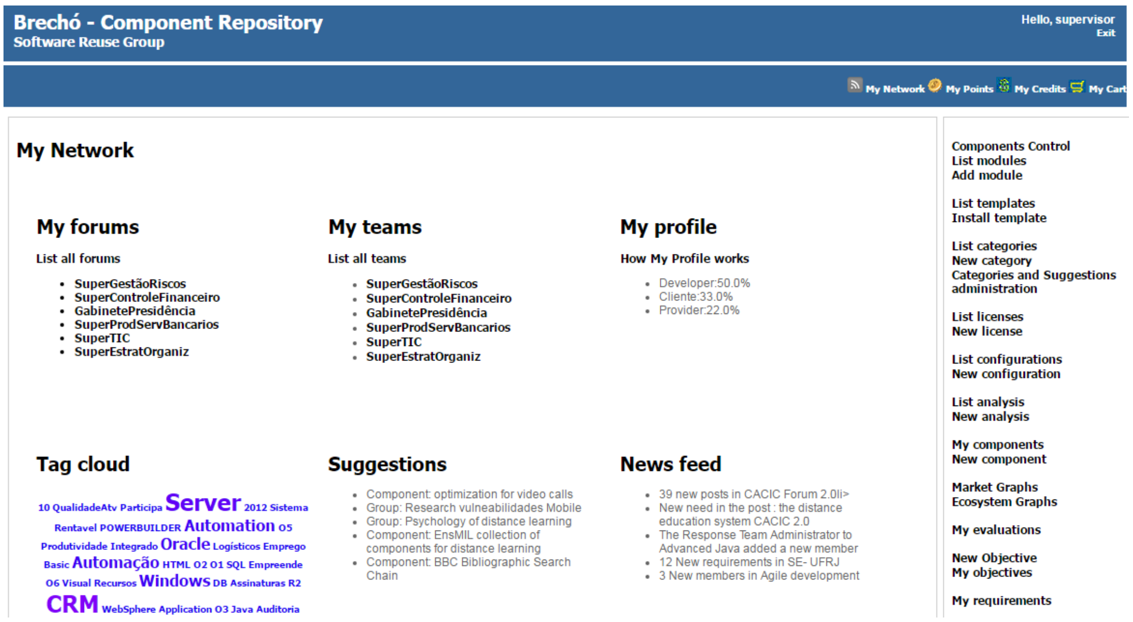


Figure 1. Brechó-SocialSECO's conceptual model


Brechó - Component Repository
Software Reuse Group

Hello, supervisor
Exit

My Network · My Points · My Credits · My Cart

My Network

My forums

List all forums

- SuperGestãoRiscos
- SuperControleFinanceiro
- GabinetePresidência
- SuperProdServBancarios
- SuperTIC
- SuperEstratOrganiz

My teams

List all teams

- SuperGestãoRiscos
- SuperControleFinanceiro
- GabinetePresidência
- SuperProdServBancarios
- SuperTIC
- SuperEstratOrganiz

My profile

How My Profile works

- Developer:50.0%
- Cliente:33.0%
- Provider:22.0%

Tag cloud

10 QualidadeAtv Participa Server 2012 Sistema Rentavel POWERBUILDER Automation 05 Produtividade Integrado Oracle Logísticos Emprego Basic Automação HTML 02 01 SQL Empreende 06 Visual Recursos Windows DB Assinaturas R2 CRM WebSphere Application 03 Java Auditoria

Suggestions

- Component: optimization for video calls
- Group: Research vulneabilidades Mobile
- Group: Psychology of distance learning
- Component: EnsMIL collection of components for distance learning
- Component: BBC Bibliographic Search Chain

News feed

- 39 new posts in CACIC Forum 2.0li>
- New need in the post : the distance education system CACIC 2.0
- The Response Team Administrator to Advanced Java added a new member
- 12 New requirements in SE- UFRJ
- 3 New members in Agile development

Components Control
List modules
Add module

List templates
Install template

List categories
New category
Categories and Suggestions administration

List licenses
New license

List configurations
New configuration

List analysis
New analysis

My components
New component

Market Graphs
Ecosystem Graphs

My evaluations

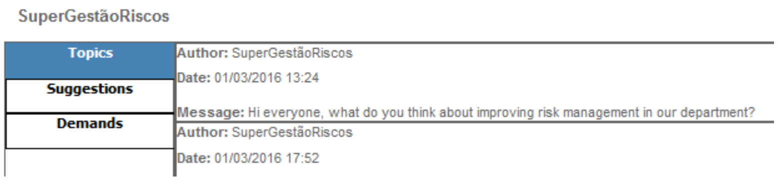
New Objective
My objectives

My requirements

Figure 2. "My Network Panel" at Brechó-SocialSECO infrastructure

3.1. Artifact/Team Forum (demand registering)

The resource *forum* implemented in Brechó-SocialSECO is organized in three sections (Figure 3). The first section is used for general discussion, organized by 'topics', that are theme-free, i.e., users can talk to IT managers and architects about an artifact's function, ask for help, report a bug etc. This section allows a potential user to communicate with others before suggesting new demands. The second section intends to gather 'suggestions' from the organizational units. Anyone who is a member of a forum can contribute with suggestions for new demands, and a forum representative is responsible for managing these suggestions, e.g., department head. Finally, the third section allows a unit representative to officially register ‘demands’. He/she registers these demands, or he/she can select a suggestion from the second section and make it into a demand.


SuperGestãoRiscos

Topics

Suggestions

Demands

Author: SuperGestãoRiscos

Date: 01/03/2016 13:24

Message: Hi everyone, what do you think about improving risk management in our department?

Author: SuperGestãoRiscos

Date: 01/03/2016 17:52

Figure 3. Forum’s discussions and sections

Users can ‘follow’ a registered demand since it involves the SECO community’s discussions and artifacts’ features. In addition, every message/suggestions are subjected to an evaluation system. An actor can vote for positive (+1) or negative (-1), regarding his/her opinion for each message. An actor can collect points for registering a suggestion that has received many positive votes. These points are used to leverage a unit’s collaborative level, as well as to reward users. Then, IT managers and architects can better understand the relevance of suggestions of the units, as well as all the demands emerging from the community. This extra information allows an IT management team to make better decisions on the selection and prioritization of demands, not only relying on the supplier’s word, but on the community’s ‘word of mouth’.

3.2. Team Creation

Actors can create teams and add other actors. Different types of teams can be created, e.g., ‘iOS users’, ‘Financial Sector’, ‘CRM extensions’ etc. Teams are defined as a type of user, which means that they keep a user profile. They can request demands, publish messages in forums, and evaluate artifacts on behalf of the team. Every team has administrators and members, and has a forum associated with it.

3.3. Actor/Artifact Profile (roles)

The SECO’s actors have many possible roles. There are internal and external actors playing as competitors, suppliers, users etc. (Lima et al., 2015). In a software acquirer, users and teams can check the proportion of actions they have performed within the SECO, being a producer, a consumer, or a simple user profile. This proportion is calculated according to their actions, such as publishing, downloading, or evaluating.

3.4. Trend Topics (tag cloud)

Tag clouds improve the way new trends and popular information are visualized in the SECO platform. It is a direct summarization of what is being discussed and evaluated by the community. The tag cloud consists of a data mining function that uses forums discussions, recommendations and demands, as well as teams’ information as inputs.

3.5. Suggestions and Recommendations (news feeds)

SECO platform can give an actor some suggestions and recommendations (news feeds). It is based on the forums he/she participates, as well as the profiles and demands he/she follows. Any forum member can post suggestions. Recommendations refer to teams, forums and/or artifacts of interest, also including information about the teams that an actor participates and new releases of artifacts. They aim to bring new information to an actor (motivating to keep him/her updated), and search for further information. From the 'team' resource, actors can send requests to take part in an existing demand.

3.6. Demand and Solution Analysis

In order to organize a given SECO, a module for exploring and manipulating networks was developed and integrated with Brechó-SocialSECO. Gephi (https://gephi.org/) was chosen due to its support to statistics algorithms, integration with Brechó's technologies (Java/MySQL) and IDEs, community support, and ease of use. This module is named SECO-DSA (Santos, 2016) and is responsible for importing SECO elements from Brechó-SocialSECO's database (SECO network graph's nodes), as well as their different types of relationships (SECO network graph's edges). The SECO elements are artifacts (APP), technologies (TEC), demands (DEM), candidate solutions (CAN APP), and business objectives (OBJ). Since the most important TECs and OBJs are those that have more artifacts that depend direct/indirectly on them, and whose artifacts have a high number of licenses, we define our graphs as shown in Figure 4. In this work, a license is the authorization to use an asset, describing rights and limitations. Also, the asset's acquisitions are tracked using licenses.

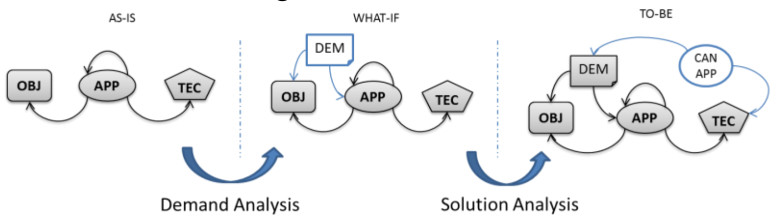
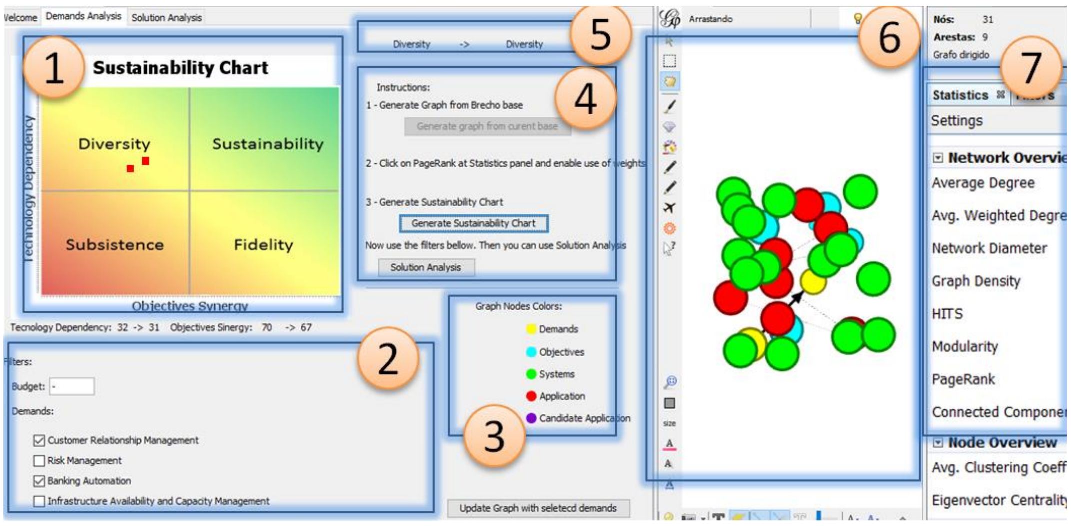


Figure 4. SECO elements at SECO-DSA Module

SECO-DSA considers three SECO platform configurations. *AS-IS* configuration presents the current artifacts, technologies and objectives in the acquirer's asset base. The *demand analysis* activity generates the *WHAT-IF* configuration based on simulation of new configurations after selecting potential demands. Finally, *TO-BE* configuration is produced by the *solution analysis* activity, in which candidate solutions are selected for each demand. All the relationships illustrated in Figure 4 are *dependencies* among SECO elements following the arrow's direction. For example, if selected, a demand X will affect artifacts Y and Z, and it will satisfy objectives A and B. Dependencies are weighted by the number of licenses carried from the origin node in this proposal.

The main goals of SECO-DSA are: (1) *to monitor objective synergy* (OBJ-SYN): allows IT managers and architects monitoring to which extent the existing artifacts in the asset base meet the acquirer's business objectives, before and after they perform *demand* and *solution analyses*; (2) *to monitor technology dependency* (TEC-DEP): allows IT managers and architects monitoring to which extent the existing artifacts depend on the technologies currently adopted by an acquirer, also before and after they perform *demand* and *solution analyses*; and (3) *to analyze the SECO platform*: use a Gephi plug-in to analyze how balanced the asset base is, i.e., how the level of sustainability of a SECO is. The metrics OBJ- SYN and TEC- DEP are measured in a 0-100% scale and allow creating a sustainability chart, as shown in Figure 5.1.

The chart indicates the sustainability status in four proposed quadrants: (a) *subsistence*: the acquirer is highly dependent on few technologies, and few objectives are satisfied by several artifacts at the same time; (b) *diversity*: there is a balanced dependency on the adopted technologies, but most of the artifacts do not meet many objectives; (c) *fidelity*: the acquirer has high dependency on a small set of technologies, but most of the artifacts are contributing to the objectives; and (d) *sustainability*: it is the ideal situation where an acquirer has low technology dependency and high objective synergy. As exemplified in Figure 5, SECO-DSA workflow begins creating a graph that represents the current asset base registered at Brechó-SocialSECO and then calculates OBJ-SYN and TEC-DEP in order to get the values for the chart's axis. Next, the user can simulate the WHAT-IF configurations when demands are chosen and the SECO network graph is updated, followed by the chart's update. Once the user is satisfied with the selected demands, he/she can select candidate solutions through the same process, but using the filtering tab to choose the candidate solutions for each selected demand.



Instructions:
1 - Generate Graph from Brecho base
2 - Click on PageRank at Statistics panel and enable use of weights
3 - Generate Sustainability Chart
Now use the filters bellow. Then you can use Solution Analysis

Tecnology Dependency: 32 -> 31 Objectives Sinergy: 70 -> 67

Filters:
Budget: -
Demands:
- [x] Customer Relationship Management
- [ ] Risk Management
- [x] Banking Automation
- [ ] Infrastructure Availability and Capacity Management

Nós: 31
Arestas: 9
Grafo dirigido

Figure 5. SECO-DSA Module as a Brechó-SocialSECO's Gephi plug-in

The main functions of the SECO-DSA panel are illustrated in Figure 5: (1) *sustainability chart*: is produced as a result of the values of OBJ- SYN and TEC- DEP for current SECO configuration and for the selected demands/solutions; (2) *filtering options*: allow users to select demands/solutions and update the network graph (inserting/removing nodes); (3) *node colors*: reserved area to explain the nodes' types; (4) *instructions and actions buttons*: present the instructions and buttons to guide demand/solution analysis, also (7); (5) *chart information:* shows information about the previous and current charts and notifies the user about the sustainability change caused by demand/solution selections; and (6) *graph visualization*: is a native Gephi's panel and shows the graph built from the asset base registered at Brechó-SocialSECO and also after user's selections. The outputs of SECO-DSA module support IT managers and architects to get insights from the asset base and to better choose demands and solutions.

4. Evaluation

In order to capture potential users' perceptions and suggestions about our infrastructure, experts in IT management activities conducted an initial feasibility study. It was based on the execution of tasks, as well as the fulfillment of feedback questionnaires. The participants should perform activities as IT managers evaluating possible demands and as architects assessing available solutions.

4.1. Planning

For this study, ten participants were invited. All of them worked in different private and public companies and had experience in the software industry and academia. An online questionnaire was applied with four sections: (1) *Informed Consent Form*: participant's rights and responsibilities; (2) *Characterization Form*: participants' background and experience with the study's concepts and similar tools; (3) *Execution Form*: eight tasks to be executed with our infrastructure, based on real data obtained from a Brazilian large banking organization; and (4) *Evaluation Form*: a questionnaire for the participant to provide feedback about ease of use and utility of existing mechanisms. Participants who live out of Rio de Janeiro used a PC remote access software. In any case, the observer took some notes. The two study goals (G) G1 and G2 were defined accordingly to the Goal-Question-Metric (GQM) paradigm (Basili et al., 1999), as described in Table 1.

Table 1. GQM for the feasibility study

Analyze	*Brechó-SocialSECO (and SECO-DSA module)*
With the purpose of	*characterizing*
With respect to	*the impact of SECO monitoring in the software acquirer's IT management activities (G1) as well as the tool usability (G2)*
The point of view from	*IT managers and architects*
In the context of	*demand and solution analysis*

The main question (Q) was: "*Are the participants able to realize the impact of SECO monitoring in the software acquirer's IT management activities for demand and solution analysis, regarding effectiveness and efficiency?*" (Q0 for G1). In other words, how the socialization mechanisms and the asset base analyses help IT managers to monitor the SECO. This perception was measured by the answers given to the study's tasks based on the following metrics (M):

M1: *Effectiveness* measures the relation between the results and the objectives:

$$Effectiveness = \frac{number\ of\ correct\ answers}{total\ number\ of\ questions}$$

M2: *Efficiency* measures the relation between the results and the resources:

$$Efficiency = \frac{number\ of\ correct\ answers}{time\ taken\ to\ participate}$$

Questions Q1-Q10 for G2 were based on Nielsen's heuristics (Table 2). To answer the questions derived from such heuristics, a 5-point scale were adopted: Totally Disagree, Disagree, Not Agree Nor Disagree, Agree, and Totally Agree. For each question, the percentage of answers was analyzed by means of a frequency chart.

4.2 Execution

The execution was individually performed with five participants in June 2016. Two of them participated in person and the others remotely. Participants signed the informed consent form and answered the characterization form. Next, each of them received an explanation about SECO concepts, as well as a Brechó-SocialSECO tutorial. Finally, each participant received the execution form, and the evaluation form at the end.

4.3. Results

Regarding the academic education, two participants reported to hold a PhD degree, two had a Master degree and one had a Bachelor degree. Figure 6 shows the experience of each participant regarding the concepts related to the study (time in years). Each concept had at least one participant with more than 4 years of experience (up to 20 years). It also refers to the experience of each participant with related tools. Collaboration tools were the most popular (at least 10 years of experience).

Table 2. Nielsen's heuristics and questions (Nielsen, 1993)

ID	Principle	Question
Q1	*Visibility of system status*	Does the system inform what is happening?
Q2	*Match between system and the real world*	Does the system explore the user language?
Q3	*User control and freedom*	Is the system easy to interact and present clear outputs?
Q4	*Consistency and standards*	Do different situations or actions represent the same thing?
Q5	*Error prevention*	Does the project predict errors instead of using messages?
Q6	*Recognition rather than recall*	Do the screens use metaphors (instructions memorization)?
Q7	*Flexibility and efficiency of use*	Does the system meet experienced users (shortcuts)?
Q8	*Aesthetic and minimalist design*	Is the information summarized and complete?
Q9	*Help users recognize/diagnose/recover from errors*	Are problems and solutions indicated?
Q10	*Help and documentation*	Are there simple and objective manuals?

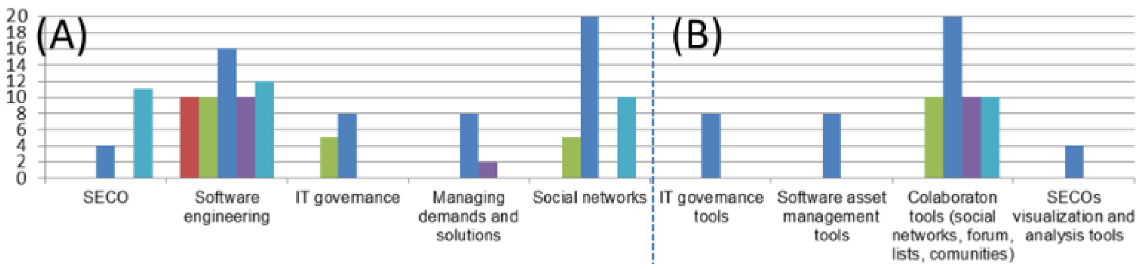


Figure 6. Participants' experience with concepts (A) and related tools (B) (time in years)

After charactering the participants' profile, results were analyzed based on their answers, duration of activity, and feedback provided in the evaluation form. The values of M1 and M2 are found in Table 3. For example, for task (2) "*For which combination of demands does the objective synergy get better?*", the participants had to simulate selecting CRM demand, Banking Automation demand, or both, and only one of them answered wrong. As shown in Table 3, the effectiveness to perform IT management activities for demand and solution analysis was positive with our infrastructure in the selected and applied context. However, the efficiency was not so high with its use. In turn, the usability measures shown in Figure 7 suggest that some mechanisms need improvements, especially because of weak evaluations obtained for the heuristics *Error prevention*, *Flexibility and efficiency of use*, and *User control and freedom*.

Table 3. Effectiveness and efficiency measures

Participant	Number of Correct Answers	Time (min)	Effectiveness	Efficiency
P1	6	65	0.750	0.092
P2	6	42	0.750	0.143
P3	5	35	0.625	0.143
P4	6	50	0.750	0.120
P5	5	51	0.625	0.098
Average	5.6	48.6	0.700	0.119

Finally, the qualitative feedback on the user's impressions was analyzed. For the question "*What were the biggest difficulties in performing the proposed tasks?*", some participants pointed out: to perform the steps required for the sustainability chart update (and to ensure that the values were also updated); to understand the sustainability moves since the points were too close and had no marks to facilitate the reading; and to find the list of members of a team. For the question "*If you are missing resources, please write* *here:*", some participants mentioned: storage of the historical series of SECO network graphs to evaluate different combinations; and automatic comparison of different SECO configurations (scenarios). Considering the question "*In your opinion, what are the most useful resources?*", some participants highlighted: sustainability chart visualization and simulations of selection's combinations; forum's structure (by specific themes and by discussions/suggestions); and trend topics mechanisms (tag cloud). Finally, the question "*Do you have any suggestion?*" allowed us to collect the following points: usability improvement for the Gephi plug-in (P2: *Richer graphical resources for analysis of the selected items, also tracking the history*) and presentation of relationships in forums (P2: *I missed graphical resources from social networks to support discussions of topics*).

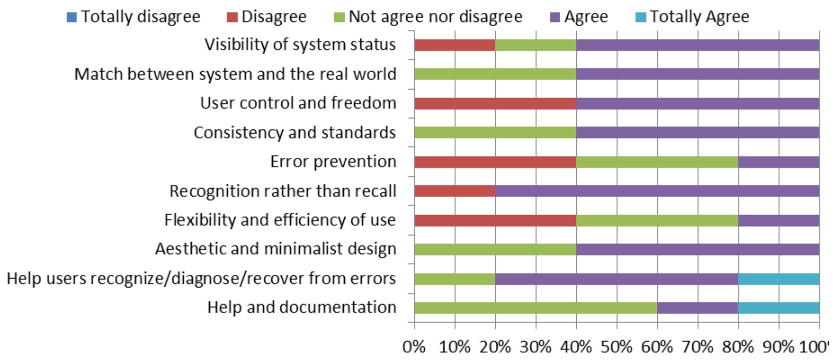


Figure 7. Usability measures

In general, the participants understood the role of our infrastructure and found the proposal useful, considering the mechanisms provided by Brechó (e.g., forum and tag cloud) and by Gephi (e.g., sustainability chart). The most popular suggestions were to improve the steps to generate the chart and to show the simulation history in order to compare and select different options. This study revealed not only corrections (e.g., bugs in button, and resetting options) but also new resources (e.g., to visualize the historic data of simulations) to be implement aiming to better monitor SECOs.

There are some limitations of our study, e.g., the low number of participants, the lack of a control group for comparisons, the lack of more and diverse scenarios, and the subjective collection of the research impressions. As threats to internal validity, we highlight: the infrastructure can influence the user's experience, and the execution form can interfere in the participant understanding. In turn, as threats to external validity, we point out: data are specific for an acquirer's reality, participants not necessarily work in the same company, and it is not possible to represent all the SECO context in a scenario.

5. Final Considerations

Current business environments and communities are continuously evolving. This reality contributes to changes in the organizational processes, affecting business objectives and demands/solutions selection. This fact motivated us to research how SECO perspective affects IT management activities, more specifically demand and solution analysis, aiming at helping IT managers and architects to make acquisition decisions. This paper presented an infrastructure to support socialization, monitoring and analysis of SECO named Brechó-SocialSECO. The asset base's socialization and analysis mechanisms were built upon a component library to aid the SECO monitoring. The main motivation is the lack of initiatives that combine market data with those obtained from the software asset database, as reported in a literature review on SECO (Manikas & Hansen, 2013). To evaluate our proposal, we conducted an initial feasibility study to observe the impact of SECO monitoring in the software acquirer's IT management activities as well as the tool usability. As a result, the effectiveness to perform IT management activities for demand and solution analysis was positive with our infrastructure, in the selected and applied context. The same happened for the most usability concerns. However, the efficiency was not high with the use of the tool.

In summary, our study brings initial indications that our proposal helps IT management teams in their daily activities. Our contributions are: (a) implementation of an infrastructure for socialization, monitoring and analysis of SECO, as well as the sustainability indicators that are critical for IT management activities; and (b) evaluation of some mechanisms of our proposal with practitioners. Future work involves a complete evaluation of Brechó-SocialSECO. The level of collaboration resulted from the implemented social mechanisms, as well as the relevance of suggestions turned into demands will be investigated. Despite several limitations of this work, we believe that a long road is right ahead and the topic can contribute to the Software Engineering area regarding the treatment of business and social challenges discussed by the research and industrial communities.

Acknowledgement

We thank CAPES and CNPq (Proc. No. PDJ 150539/20016-9) for the financial support.

References

Basili, V.R., Shull, F., Lanubile, F. (1999) "Building Knowledge through Families of Experiments". IEEE Transactions on Software Engineering 25(4):456-473.

Finkelstein, A. (2014) "Rethinking Software: Business Change and the Consequences for Software Engineering", In: 22nd IEEE RE, Keynote, Karlskrona, Sweeden.

Gartner (2007) "Evaluating Global-Class Software Ecosystems". Available at: <https://www.gartner.com/doc/506608/evaluating-globalclass-software-ecosystems>.

Goeminne, M., Mens, T. (2010) "A Framework for Analysing and Visualising Open Source Software Ecosystems", In: Joint ERCIM EVOL & IWPSE, Antwerp, Belgium, 42-47.

Lima, T., Santos, R., Werner, C. (2015) "A Survey on Socio-Technical Resources for Software Ecosystems", In: 7th ACM MEDES, Caraguatatuba, Brazil, 72-79.

Manikas, K., Hansen, K.M. (2013) "Software Ecosystems – A Systematic Literature Review". The Journal of Systems and Software 86(5):1294-1306.

Nielsen, J. (1993) "Usability Engineering". Cambridge: Academic Press.

Pérez, J., Deshayes, R., Goeminne, M., Mens, T. (2012) "Seconda: Software Ecosystem Analysis Dashboard", In: 16th CSMR, Szeged, Hungary, 527-530.

Santos, R. (2016) "Managing and Monitoring Software Ecosystem to Support Demand and Solution Analysis". PhD Thesis, COPPE/UFRJ, Rio de Janeiro, Brazil, 246f.

Seichter, D. *et al.* (2010) "Knowledge Management in Software Ecosystems: Software Artefacts as First-class Citizens", In: 4th ECSAW, Copenhagen, Denmark, 119-126.

Werner, C. *et al.* (2009) "Towards a Component and Service Marketplace with Brechó Library", In: IADIS International Conference on WWW/Internet, Rome, Italy, 567-574.

Experience Report and Challenges for Systems-of-Systems Engineering: A Real Case in the Brazilian Defense Domain

Carlos Eduardo de B. Paes[1,2], **Valdemar Vicente Graciano Neto**[2,3,4], **Flávio Oquendo**[3], **Elisa Yumi Nakagawa**[2]

[1] Pontifical Catholic University of São Paulo
São Paulo – SP – Brazil

[2]ICMC – University of São Paulo
São Carlos – SP – Brazil

[3]IRISA-UMR CNRS – Université Bretagne Sud, Vannes, France.

[4]Instituto de Informática – Universidade Federal de Goiás (UFG)
Goiânia – GO – Brazil

carlosp@pucsp.br, valdemarneto@inf.ufg.br

flavio.oquendo@irisa.fr, elisa@icmc.usp.br

Abstract. *Defense domain is a crucial branch of the government of any country since it ensures the national security and supremacy. Hence, it is imperative that the technologies adopted that support their operations must be aligned with the cutting-edge scientific research results and technologies. In particular, software systems play a crucial role in defense domain, since they are usually connected among themselves, forming alliances known as System-of-Systems (SoS). In the last decades, SoS conception has evolved and Brazil has also adopted SoS' development strategies. However, we identified a lack of reports communicating challenges faced and strategies adopted to carry out SoS Engineering (SoSE) in Brazil. In this direction, this paper reports experience, results, and challenges of a real project carried out in Brazilian Department of Defense, particularly in the Navy's context. We report the conception of an SoS named Blue Amazon Management System (SisGAAz), a SoS for Brazil's defense and maritime surveillance. We claim that this report contributes to SoSE since it offers a panorama of SoS conception in Brazil, representing, as a matter of fact, the state of the art in SoSE for Defense in our country, a paramount artifact for the advance of SoS research.*

1. Introduction

Software-intensive Systems-of-Systems (SoS)[1] are strategic. They have been developed and adopted as a first class citizen in a range of domains, particularly in defense domain. Many systems have been combined, supporting reliable and trustworthy operations performed by army, navy, and aeronautics, delivering complex functionalities by means of defense SoS, and exhibiting crucial emergent behaviors, such as, the defense of a country itself as a result of the interoperability among all those individual systems. In this sense,

[1]For sake of simplicity, SoS will be used interchangeably to express singular and plural.

SoS Engineering (SoSE) has become strategic, guiding how to develop constituent systems, and offering insights on how to interoperate them to accomplish complex missions.

For a long time, interoperable systems in the defense area were largely used as instances of SoS. However, a small focus has been given on reports of challenges, problems, and limitations identified during the SoSE development life cycle. Those lessons learned are important since they represent expertise knowledge and experience that could be shared to aid the conduction of other SoSE development projects, in military domain or even other civil domains. Such lessons can foster the development of SoSE research, exposing the gaps, challenges and difficulties, to point out directions to which the investigation efforts should be invested. Unfortunately, this knowledge is often restricted to project documents in defense departments, and, particularly, there is a lack of reports of SoSE projects, specially in Brazilian defense area.

In this direction, the main goal of this paper is presenting an overview on the current practice on SoSE in Brazil, externalizing the challenges faced during the conduction of a real project, and arising challenges that still must be overcome. In particular, we perform an analysis of the challenges identified in the early stages of the SoSE life cycle (Concept of Operations, Requirements Engineering, and Architecture Design), carried out in a real project of the defense area. In this paper, we analyze SisGAAz, a Brazilian Navy Management SoS project composed of a set of interoperable systems to collect, share, analyze, display operational information, and provide decision support regarding the Blue Amazon. Many of these systems are already in use by the Brazilian Navy and others will be developed and integrated during the systems development [Chaves 2013]. The main contribution of the paper is exposing those challenges, opening research opportunities for SoSE in Brazil. The remainder of this paper is organized as follows. Section 2 presents a background. Section 3 presents an overview about SisGAAz project. Section 4 outlines challenges, limitations and problems identified during the first phase of the project. Section 5 points out for advancements we should conduct in order to overcome the highlighted challenges. Finally, Section 6 presents final remarks.

2. Foundations

SoS result from other operationally independent systems (so-called constituents) working together to reach common goals. Each constituent performs its individual mission contributing to the success of the global missions [Silva et al. 2014]. SoS have been consensually distinguished by a set of inherent characteristics [Maier 1998]: (i) operational independence of the constituent systems, since constituents have their own operation even if they work within the scope of the SoS; (ii) managerial independence of the constituent systems, which can be independently managed by distinct organizations and stakeholders; (iii) evolutionary development of the SoS, since the SoS evolve as a consequence of the individual evolution of its constituents, their functions and purposes; (iv) distribution, since constituents' interoperability relies on some communication technology, and (v) emergent behavior, which enables the SoS to provide new functionalities from interactions among constituent systems and that are not localized in a single constituent. In particular, missions are an important SoS concept, specially in military SoS.

SoS are categorized according to its degree of authority that it has over the constituent systems that cooperate to the accomplishment of the missions

[Department of Defense 2008]. In this perspective, four basic categories of SoS can be defined: (i) Directed – The SoS is controlled by a central entity and it is designed and operated for fulfilling specific purposes. Constituent systems can have their operational and managerial independence, but their behavior is subordinated to the central control and its purposes; (ii) Acknowledged – Goals, resources, and central control of the SoS are all recognized, but the constituent systems retain their independent management and their behavior is not subordinated to the central managed purpose; (iii) Collaborative – Constituent systems voluntarily collaborate in a greater or lesser degree in order to address shared or common interests. In this case, a central control has little coactivity over the behavior of the constituent systems and it typically offers standards to allow the collaboration among those systems; and (iv) Virtual – There is no central control and universal purposes, and such purposes are neither designed or expected in many cases, so that constituent systems operate in a distributed and uncoordinated environment where the mechanisms to maintain the SoS are not evident.

For a long time, SoS modeling has been carried out under a traditional document-centric perspective, suffering of (i) replication of information, (ii) lack of traceability between documents, (iii) inconsistencies of information and business rules, and (iv) difficulties to handle and search information in such documents. However, over the past decade, SoS engineers have significantly increased the adoption of Model-Based Systems Engineering (MBSE), a SoSE approach that shifts from a document-centric perspective for a model-based reality, emphasizing the development and adoption of models in SoSE [Ramos et al. 2012]. MBSE promises (i) a more effective knowledge management that can enhance the ability of stakeholders to understand the system and its behavior and performance, (ii) enhanced team communications, (iii) explicit processes for reasoning about system issues, (iv) early detection of errors and omissions, (v) improved systems architecture, (vi) detailed design integrity, and (vii) effective design traceability [Kalawsky et al. 2013, Do et al. 2014]. SysML[2] (System Engineering Modeling Language), an extension of UML[3] (Unified Modeling Language), is considered a central standard notation in MBSE [Industries Alliance 2003, Delligatti 2013].

3. SisGAAz Overview

The Brazilian Navy has developed a system called SisGAAz to meet the guidelines of the National Defense Strategy of Brazil for managing an area known as Blue Amazon. The Blue Amazon is associated with an oceanic area and corresponds to Brazilian Jurisdictional Waters (BJW), international areas of responsibility for the Search and Rescue operations (SAR - Search and Rescue) and areas of specific interest that go beyond the BJW and the SAR area [Chaves 2013].

SisGAAz supports Brazilian Navy's activities in its constitutional allocations and subsidiaries assignments, such as protecting the national frontiers and ensuring the brazilian sovereignty. The project involves a diverse team with several professional roles such as Telecommunication Engineers, Systems Engineering, Software Engineers, Domain Experts, Project Managers, Quality Managers, Information Managers, and Infrastructure Technicians, totalizing around 30 people involved.

[2]http://sysml.org/

[3]http://www.uml.org/

SisGAAz's mission is *monitoring and controlling, in an interoperable way, the national maritime area (waters under brazilian jurisdiction), international areas of responsibility for the search and rescue operations, and areas of specific interest that go beyond the previous ones, in order to contribute to the strategic mobility, represented by the ability to respond promptly to any threat, emergency, aggression or illegality*. SisGAAz's main purposes are: (i) information sharing and interoperability among the institutions with interest in the sea (national and international, public and private); and (ii) supporting network centric operations and the providing decision support by a shared virtual environment. Figure 1 depicts the government entities that maintain relations and operations with the Brazil's Navy that benefit from the SisGAAz. Entities are illustrated as actors and include several ministries, Army, and Justice.

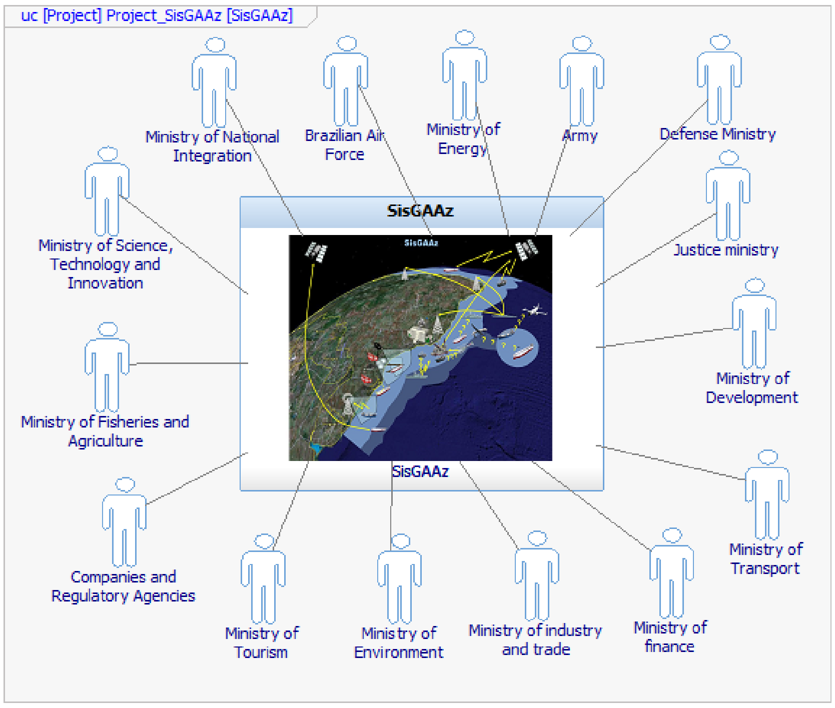

uc [Project] Project_SisGAAz [SisGAAz]
Ministry of National Integration
Brazilian Air Force
Ministry of Energy
Army
Defense Ministry
Ministry of Science, Technology and Innovation
SisGAAz
Justice ministry
Ministry of Development
Ministry of Fisheries and Agriculture
SisGAAz
Ministry of Transport
Companies and Regulatory Agencies
Ministry of Tourism
Ministry of Environment
Ministry of industry and trade
Ministry of finance

Figure 1. SisGAAz external entities and stakeholders.

SisGAAz development was divided in two phases: conceptualization and development. The conceptualization phase involved the system specification and design (operational concept and requirements), elaboration of project management plans, and architecture design. This phase was completed in two years and it was developed in the following four tasks: (i) project management planning; (ii) elaboration of Concepts of Operation (ConOps); (iii) elicitation, analysis, and documentation of system requirements; and (iv) conception and design of SoS architecture. Figure 2 illustrates the project life cycle.

The development of SisGAAz follows an evolutionary approach (iterative and incremental). Initially, a small core of functionalities is already implemented and deployed. Subsequently, the implementation activities are performed in a way that new functionalities are added to constituents according to the local demands of an operational area. For example, the Navy district of Rio de Janeiro is an operational area, as it is São Paulo. Rio de Janeiro district hosts the first constituents implemented according to the local de-

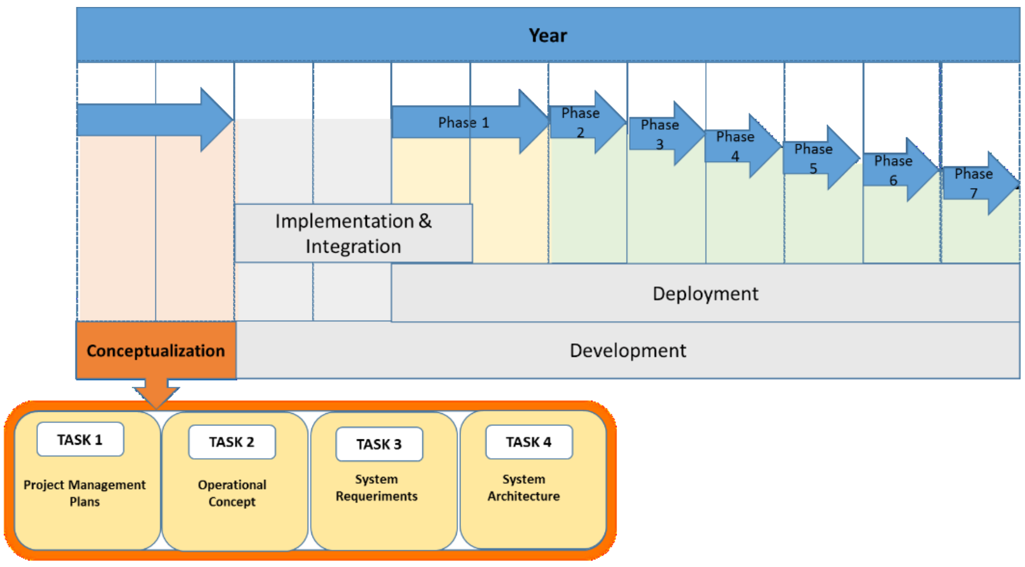


Figure 2. SisGAAz Project's Life cycle

mands. After that, the implementation will take place São Paulo. The new constituents developed there matches their specific demands, and after developed, they are joined to the first ones to form the preliminary body of the SoS (with constituents from São Paulo and Rio de Janeiro). Next, this work is iteratively carried out in other regions of Brazil, creating new constituents, connecting them to the others, realizing the SoS, and expanding the SiSGAAz SoS progressively.

The proposed SoS has five types of constituents, each one responsible for one of the following assignments: logistics support, cybersecurity, intelligence, command and control (C2), communications, and remote sensing. Logistics constituent aims to provide support to logistics operations regarding Navy's equipment, such as acquisition and transport of material to national oceans supervision. Cybernetics constituent provides capabilities related to SisGAAz safety and security aspects, supporting (i) availability and assurance of confidentiality of information and services, (ii) defense and protection of networks (internal and external), (iii) monitoring the full spectrum of cyber warfare operations, and (iv) safety and protection of services and constituents that forms the SisGAAz. The command and control (C2) constituents are military systems that provide features to support planning of command operations and control of constituents in the accomplishment of the mission, playing a type of central authority role. The communications constituents provide the technological infrastructure to integrate the various military organizations and SisGAAz constituents. The intelligence constituent aims to support planning and execution of operations, according to the ConOps. Finally, the remote sensing constituent provides functionality for collecting, processing and distributing data from a variety of sensors, new or legacy, which make up the SisGAAz and data received through cooperation with other organizations. Each brazilian geographic area can host a subset of those types of constituents that are linked together to form the national scale SoS known as SisGAAZ.

4. A Panorama of SoSE in Brazil

After our participation in the first phase of SisGAAz project, we glimpsed a panorama about how SoS is currently developed in Brazil. Next, we distill our view on SoSE in Brazil under some remarkable perspectives.

4.1. SoS characterization

Most of current military systems are part of a SoS, even if not explicitly recognized as such. Although SisGAAz project has focused on the development and acquisition of independent systems, many of those systems have been and will be created and integrated into the SisGAAz without explicit consideration of SoS type. In a first perspectice, SisGAAz could be considered as a directed SoS, specially for the presence of a C2 constituent with some degree of central authority. However, SisGAAz's constituents share objectives, management, and independent resources. They are, in practice, a collaborative SoS, but (i) they have not been thought as such, (ii) other constituents have been arbitrarily added to such SoS, and the level of control is not clear, which reflects on how the SoS is engineered [Dahmann and Baldwin 2008]. In SisGAAz project, those issues and SoS characterization were not considered. This issues impact directly in the dynamics of project management, systems engineering and architectural decisions in the system.

4.2. System Engineering Process

During the conceptualization phase of SisGAAz, a traditional document-centric system engineering process was adopted. Such process is based on American Department of Defense (DoD) MIL-STD-498, a process to guide how to perform activities for defining the SoS system architecture [Department of Defense 1994] and standardize software development and its documentation. It proposes different types of documents (known as Data Item Descriptions (DIDs)) to elaborate the project plans, concept of operations, requirements, design, test, software, support manual etc. Such approach exhibits important limitations that make the SoS development process costly, bureaucratic, and heavy, sharing all of the aforementioned drawbacks brought by document-centered approaches. Specific issues problems due to (i) information spread across several documents, (ii) lack of correspondences between requirements, analysis, and system design, becoming them hard to assess, (iii) difficulties of comprehension of the whole SoS and its architecture, (iv) difficulties to maintain such disjoint set of artifacts, what significantly impacts the total cost and effort, and (v) obsolescence of the SoS architecture [Industries Alliance 2003]. Indeed, even with the high cost, the artifacts become inconsistent and obsolete [Friedenthal et al. 2008, Delligatti 2013].

4.3. System Engineering Lifecycle Management

SisGAAz involves several professionals from different knowledge areas distributed in two states (São Paulo and Rio de Janeiro). The communication between the team members was realized by e-mails, phone and files exchanged through a repository of version control. As a consequence, we have identified some problems: (i) excessive number of documents and files stored in a huge repository; (ii) fragmented records of project information scattered in numerous emails; (iii) problems to share the information between the engineers during the project; and (iv) loss of important information during the meetings by telephone. This poor and ineffective communication brought a negative impact on the progress of the project and mainly for the SoS architectural specification, particularly in this real case, in which the project is large, complex and multidisciplinary. The system engineering life cycle was realized without an integrated and collaborative system-engineering environment. Consequently, the team did not have a clear view of the system engineering process used in the project and it was very difficult to realize the tasks and activities proposed to the project.

4.4. Models

Some system engineering models were developed during the architectural system design step. Business Process Model and Notation (BPMN)[4] and Data Flow Diagram (DFD) were used for the representation of operational processes. A combination of UML and SysML models were adopted for materializing the System and Subsystem Design Description (SSDD). Many models (BPMN, SysML and UML) created during the project used incompatible elements of language to represent the SiSGAAz architectural description. This combination produced incomplete and inconsistent models according to the syntax and semantics proposed by the language. This is a result of the lack of an appropriate MBSE method to support the correct use of SysML. Distinct modeling approaches can cause a lack of standardization during the development phase. In addition, models in SysML and UML are largely static and descriptive and do not capture the dynamics and executable aspects of emergent behaviors of SoS.

5. Challenges and Advancements

As reported, there are types of limitations in the way SoS are currently engineered in Brazil. As a matter of fact, the SoSE practice in Brazil has been divergent from the world-wide state of the art and practice that we yielded previously. Thus, it is paramount to take those observations as motivations to research and advance in our practice to produce SoS. In this direction, we argue that each of the aforementioned key viewpoints configure specific challenges raised. In this direction we raise the following challenges (C) for the current practice of SoSE in our country:

C1. Conception of SoSE processes that are aligned with Maiers' taxonomy: Current SoSE practice in Brazil does not take into account the aforementioned taxonomy [Maier 1998]. Hence, it is important to adopt and conceive SoSE processes that aides the stakeholders involved to discern the nature of the SoS being produced, and in which the activities support the realization of the SoS architecture, avoiding architectural degradation [Gurgel et al. 2014]. Hybrid cases, i.e., those ones in which multiple types of central authority can co-exist, as in SiSGAAz, must also be investigated;
C2. Substitution of Ancient Processes: Current practice in Brazil still adopts document-centric approaches. In alignment with international trends, SoSE processes must be migrated to modern proposals;
C3. Absence of Process Standardization and Automation: Even with the adoption of some industrial standards, there is a lack of standardization in the processes. The process recommendations are not strictly followed. Hence, it is necessary to propose some methods to facilitate the adherence of the stakeholders to the processes elaborated and/or adopted and their institutionalization. Software Process Improvement practice can

[4]http://www.omg.org/spec/BPMN/2.0/

give some guidance on this direction and Governance politics must also be proposed for this scenario;
C4. Models and Automation: It is necessary to shift the SoSE practice in Brazil from the outdated document-centric approaches to the cutting-edge technologies and processes for SoSE, in particular MBSE. There is a lack of models in the current approaches, a low level of abstraction in the documents, and a low possibility to automate SoSE production activities tasks. All this scenario increases the complexity in the development, reduces the maintainability and traceability of the final product, and causes a lack of sync between the final product and the correspondent documents. Solutions must be adopted, extended, and proposed to tame those drawbacks.

Since we identified those challenges, we also propose some intervention proposals that can aide the advancement of current practice of SoSE in Brazil. Among these practices, we can highlight the following:
SoS Processes (C1, C2, C3): we noticed a lack of guidance to support the SoS characterization and processes that reinforce the adherence of team members to their tasks. New processes specially tailored for SoS have been proposed [Goncalves et al. 2015]. They must be adopted and/or extended to overcome such difficulties and to support such processes institutionalization and a strict adherence to SoS taxonomy;
Conception of Collaborative and Distributed Software Development Environments for the conception of SoS (C1, C2, C3): Engineering SoS involves a range of people that inherently collaborates to deliver a final product. Specially in SoS, there is a high degree of diversity of the stakeholders and institutions involved in the production life cycle, many times geographically spread. Hence, it is paramount to develop software environments to support a suitable management of the SoS development with all sort of complexity involved. Those tools should support collaborative work with an additional dimension of global distributed development, supporting a current reality of Distributed Software Development [Marczak et al. 2014, Santos et al. 2015], a reality in SoS conception. Hence we propose that collaborative web-based environments must be suitably conceived in order to match the modern demands imposed by SoS production;
Endorsement of models, metamodels and MBSE approaches in SoSE (C4): Models and metamodels foster an institutionalized practice of documenting software, even in SoS context, since they become first-class citizen in the production chain [Graciano Neto et al. 2015], becoming a golden feature in software and systems development. MBSE approaches and processes for SoS are imperative, since models contribute to automation, traceability, abstraction. They must become the *de-facto* standard to drive SoS conception and specification, overcoming the drawbacks deeply related to document-centric approaches, and adopting the cutting-edge technologies recently proposed to skip the ancient frontiers of SoS production. Remarkable examples include the use of architectural description languages (ADL) for documenting SoS software architectures, such as the recently proposed SoSADL [Oquendo 2016];
Simulation technologies (C4): Simulations are a well-recognized technique in SoSE [Graciano Neto et al. 2014]. Since it offers a model that facilitates the visualization of the SoS operation before properly being deployed, it anticipates the detection of deffects and problems promoting an early correction of them, provides a dynamic view for SoS architectural description models to aide SoS architects in their activities in the software architecture lifecycle, facilitates validation and verification tasks [Michael et al. 2011],

specially tackling dynamic properties inherent to SoS, such as emergent behaviors [Graciano Neto and Nakagawa 2015]. A remarkable example is DEVS (Discrete Event System Specification), a modeling formalism for SoS that relies on sequence diagrams, state diagrams, ports specification, and input/output events [Zeigler et al. 2012].

These are important directions for the progress of SoS engineering practice and improvement of systems engineering industry in Brazil. We assume that each of these factors creates limitation and problems for SoS engineering. Overcoming them is imperative, and the brazilian SoS community can contribute in order to extend international instances of SoS production, proposing solutions and advancing the state of the art and state of the practice in Brazil to ground a better future of technology to our country, specially regarding the national security and supremacy.

6. Final Remarks

SoS have emerged and SoS engineering (SoSE) has become the heart of the development of such systems. This paper presented a big picture of the practice in SoSE in Brazil. Our main contribution is presenting an experience report of a real project conducted in Brazilian's defense context, externalizing the state-of-the-practice currently followed in Brazil. We believe that the SoS projects for defense areas need to incorporate new practices and approaches to system engineering and architecture design. Nowadays the systems engineering has been undergoing a major transformation that consists in the adoption of consolidated practices from modern software engineering to SoSE. We presented the challenges that we will face in SoSE in Brazil in the next years, and we propose some directions to support a sustainable path to realize such improvements that are in alignment with other recently proposed research directions [Graciano Neto et al. 2016]. We are aware of other international initiatives on reporting experience in SoSE [Do et al. 2014]. However, from the brazilian perspective, this is an innovative artifact. Future work include working on some of the branches we raised, and extending our reports in other directions, such as externalizing the software architectural description of SiSGAAz, and adopting such SoS as case study scenarios for solutions we have worked on in the research groups we are enrolled (START/ICMC and ArchWARE/IRISA).

Acknowledgments We thank PIPEq/PUC-SP, CNPq (grant:201230/2015-1/SWE) and FAPESP (grant: 2014/02244-7).

References

Chaves, S. F. A. (2013). Blue amazon system management (sisgaaz): the first step towards effective control of the brazilian maritime area. Technical report, Brazilian Department of War Studies School.

Dahmann, J. S. and Baldwin, K. J. (2008). Understanding the current state of us defense systems of systems and the implications for systems engineering. In *Systems Conference*, pages 1–7.

Delligatti, L. (2013). *SysML Distilled: A Brief Guide to the Systems Modeling Language*. Pearson.

Department of Defense (1994). MIL-STD-498, Software Development and Documentation. MIL 498, DOD.

Department of Defense (2008). Systems engineering guide for system of systems. Technical report, Department of Defense.

Do, Q., Cook, S., and Lay, M. (2014). An investigation of MBSE practices across the contractual boundary. *Procedia Computer Science*, 28:692 – 701. 2014 Conference on Systems Engineering Research.

Friedenthal, S., Moore, A., and Steiner, R. (2008). *A Practical Guide to SysML: Systems Modeling Language*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Goncalves, M. B., Oquendo, F., and Nakagawa, E. Y. (2015). Towards architectural synthesis of systems-of-systems. WDES, pages 25–32, Belo Horizonte, Brazil. SBC.

Graciano Neto, V. V., Garcés, L., Guessi, M., de Oliveira, L. B. R., and Oquendo, F. (2015). On the equivalence between reference architectures and metamodels. CobRA '15, pages 21–24, New York, NY, USA. ACM.

Graciano Neto, V. V., Guessi, M., Oliveira, L. B. R., Oquendo, F., and Nakagawa, E. Y. (2014). Investigating the model-driven development for systems-of-systems. ECSAW, pages 22:1–22:8, Vienna, Austria. ACM.

Graciano Neto, V. V. and Nakagawa, E. Y. (2015). A biological inspiration to support emergent behavior in systems-of-systems development. WDES' 15, pages 33–40, Belo Horizonte, Brazil. SBC.

Graciano Neto, V. V., Oquendo, F., and Nakagawa, E. Y. (2016). Systems-of-systems: Challenges for information systems research in the next 10 years. In *GRANDSI-BR*, pages 1–3, Florianópolis, Brazil. SBC.

Gurgel, A., Macia, I., Garcia, A., von Staa, A., Mezini, M., Eichberg, M., and Mitschke, R. (2014). Blending and reusing rules for architectural degradation prevention. MODULARITY '14, pages 61–72, New York, NY, USA. ACM.

Industries Alliance (2003). ANSI/EIA 632 - Processes for Engineering a System. ANSI 632, SAE International.

Kalawsky, R. S., O'Brien, J., Chong, S., Wong, C., Jia, H., Pan, H., and Moore, P. R. (2013). Bridging the gaps in a model-based system engineering workflow by encompassing hardware-in-the-loop simulation. *IEEE Systems Journal*, 7(4):593–605.

Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering*, 1(4):267–284.

Marczak, S., Perin, M., Prikladnicki, R., and Ayub, C. (2014). On the identification of factors that promote highperformance projects in distributed development. WDES' 14, pages 17–24, Maceió, Brazil. SBC.

Michael, J. B., Drusinsky, D., Otani, T. W., and Shing, M.-T. (2011). Verification and validation for trustworthy software systems. *IEEE Software*, 28(6):86–92.

Oquendo, F. (2016). Formally Describing the Software Architecture of Systems-of-Systems with SosADL. In *SOSE*, pages 1–6, Kongsberg, Norway. To appear.

Ramos, A. L., Ferreira, J. V., and Barceló, J. (2012). Model-based systems engineering: An emerging approach for modern systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(1):101–111.

Santos, R., Junior, I. F., Lima, T., and Hernández, L. (2015). Análise histórica do wdds/wdes. WDES' 15, pages 1–8, Belo Horizonte, Brazil. SBC.

Silva, E., Cavalcante, E., Batista, T., Oquendo, F., Delicato, F. C., and Pires, P. F. (2014). On the characterization of missions of systems-of-systems. ECSAW, pages 26:1–26:8, Vienna, Austria. ACM.

Zeigler, B. P., Sarjoughian, H. S., Duboz, R., and Souli, J.-C. (2012). *Guide to Modeling and Simulation of Systems of Systems*. Springer.

A Service-Based Architecture for Virtual and Collaborative System of Systems

Rosana T. Vaccare Braga, Iohan Gonçalves Vargas and Thiago Gottardi

[1]Instituto de Ciências Matemáticas e Computação – Universidade de São Paulo
Av. do Trabalhador São-Carlense, 400 – 13560-970 – São Carlos – São Paulo

rtvb@icmc.usp.br, iohan@usp.br, gottardi@icmc.usp.br

Abstract. *System of Systems (SoS) provide means of integrating independent systems aiming at achieving goals that could not be accomplished if these systems were executed isolated. Among the several types of SoS identified in the literature, virtual SoS still pose several challenges to be built. By not depending on a central coordinator, virtual SoS are the most robust type of SoS. On the other hand, collaborative SoS presents the advantage of having a well established emergent behavior design, which we intend to combine with the robustness of virtual SoS. Therefore, in this paper we propose a reference architecture to build SoS of a mixed type between virtual and collaborative. We present a prototype we built as a proof of concept of the proposed architecture that employs services for communication among constituents. We also describe how the architecture has been applied for a SoS in the software engineering tool domain. As conclusions, the architecture was successfully applied to implement a mixed-type SoS since its inception. In addition, the developed prototype can be reused for other SoS instances in different domains.*

1. Introduction

A System of Systems (SoS) is a collection of independent systems and their interrelationships to produce a system that presents additional behavior that is greater than the sum of the parts [Boardman and Sauser, 2006]. They have great potential of obtaining advantages of several existing systems to accomplish emergent behavior. The challenge is to know how to integrate the systems to become constituent systems (CS), considering their relationships and participation in the whole SoS.

Several types of SoS have been identified in the literature. For example, Maier [1998] classifies them as directed (a centrally managed SoS built and managed to fulfill specific purposes), collaborative (a SoS where CS interact to fulfill central purposes guided by central players), and virtual (a SoS with emergent behavior that has not a central management authority).

Maier classified the World Wide Web as a well known Virtual SoS [Maier, 1998]. By not having a central management authority, this category of SoS allows any interested party to create a new CS and join the SoS. At the same time that we want systems to be independent, we need to ensure that the SoS goals are met, i.e., it is important to plan specific emergent behaviors expected to be fulfilled, similarly to a Collaborative SoS design. We cannot discern and distinguish exactly how the system functionality is achieved in a virtual SoS. Nielsen et al. [2015] consider a large-scale complex IT system as a virtual system, justified by the fact that they consist of systems that work together, maybe with mutual interest, but are owned by individual organizations that may be competing. It is important to highlight that we have not found any SoS architectures proposing solutions that satisfy the needs of such systems.

Therefore, we intend to identify how the gap between Virtual and Collaborative SoS designs could be interpolated with a Mixed-Type SoS.

Therefore, this paper has the following goals: (i) to identify a set of requirements for the Mixed-Type SoS, (ii) to present a service-based architecture, which we named MV-SoSA, that serves as a basis when composing new Mixed-Type SoS, (iii) to present an implementation of MV-SoSA that we refer as MV-SoS-Prototype, and (iv) to describe a concrete example of a Mixed-Type SoS employing MV-SoS-Prototype as part of a case study.

The remainder of the paper is organized as follows. In Section 2 we summarize the concepts of SoS and Services. In Section 3 we present the architecture proposal: first we identify the requirements for a Mixed-type SoS (Section 3.1), followed by an overview of MV-SoSA (Section 3.2). In Section 4 we present MV-SoS-Prototype. In Section 5 we describe how MV-SoSA is instantiated to develop a SoS in the software reuse domain. Related work is discussed in Section 6. Finally, the conclusions, as well as future work, are discussed in Section 7.

2. Background

Systems of Systems (SoS) allow the aggregation of independent systems, together with their relationships, to form a whole that is greater than the sum of the parts [Maier, 1998]. Thus, it is possible to achieve complex goals that would not be easily achieved individually by the CS. Emergent behaviors can also arise at any time and should be easy to model.

Maier (1998) classifies SoS based on managerial characteristics, suggesting three types of SoS: Directed, where the integrated SoS is built and managed to fulfill specific purposes and CS maintain an ability to operate independently, but their normal operational mode is subordinated to the central managed purpose; Collaborative, where the CS interact voluntarily to fulfill agreed upon central purposes and central players collectively decide how to provide or deny services; and Virtual, where there is neither a central management authority nor a centrally agreed upon purpose for the SoS, but behavior emerges naturally from the composition.

Maier [1998] has defined five important characteristics of SoS: operational and managerial independence of CS, emergent behavior, geographical distribution, and evolutionary development processes. These characteristics are relevant in the context of this work, as the proposed architecture will allow several independent systems to work together forming a SoS, whilst they can freely enter or leave the composition without having their independence affected at all.

Service-orientation has been presented as a potential mechanism to allow the construction of SoS [Madni and Sievers, 2014]. A service has a public interface that is available and inter-operable, and it can dynamically connect to other services [Mahmoud, 2005].

The use of services is appropriate in distributed software development environments, in which the integration among different tools is required, preferably in a transparent way, in particular for SoS. Indeed, in a systematic review published recently [Vargas et al., 2016], it was verified that a considerable number of publications, approximately 51.72%, have explored the use of services for integration of CS in the SoS context.

3. Architecture Proposal

In this exploratory study, we propose a generic architecture for building a Mixed-Type SoS. We have initially considered SoS in the information systems domain, where each CS can make available, through services, pieces of information that are potentially of interest to achieve the SoS emergent behavior.

By its own definition, virtual SoS might not be as reliable as required by organizations, because they could enable behaviors not predicted by the CS owners. In the same manner, organizations might also require expected behaviors that cannot be covered by a virtual SoS implementation. On the other hand, the organizations may also require that the SoS must be designed with pre-planned emergent behaviors, unlike virtual SoS, since they cannot be developed to guarantee the emergent behavior. We study a solution to this problem by introducing some characteristics extracted from the collaborative type, more specifically, we suggest that a predictable set of trusted systems is kept by each system willing to participate in the SoS. The implementation of this suggestion maintains the virtual characteristic of the SoS, because: i) CS are not obliged to provide predicted services; ii) CS do not have a coordinator or a previously planned collaboration sequence; and iii) CS have total autonomy to perform their tasks, without depending on a pre-specified choreography.

3.1. Requirements for the Mixed-Type SoS

We have established a set of requirements for the Mixed-Type SoS, listed on Table 1 and explained below. They resulted from the analysis of related works on Collaborative SoS [Mahmood and Montagna, 2012; Black and Fletcher, 2006], and on Virtual SoS [Ramos, 2014], which allowed us to provide a simple design for our Mixed-type SoS. It is important to notice that the classification of SoS is not orthogonal: according to Nielsen et al. [2015], autonomy and ownership have to be maintained, but the SoS type is not expected to be uniform within a SoS, but local subsystems of different types might arise.

#	Requirement
R_1	the proposed System of Systems should be composed by a dynamic set of CS, similarly to a virtual SoS
R_2	the CS must be independent, both from the operational and managerial perspectives
R_3	the CS can enter or exit the SoS at any time, without compromising the working of the independent systems
R_4	each CS should keep a list of trustworthy CS
R_5	each CS can provide services to both users and other trustworthy CS
R_6	components and list of trusted CS must be dynamically changeable
R_7	unique identifiers could be used to identify objects in the SoS
R_8	each CS is responsible to know its managed objects, so the authorization concern should be implemented independently from authentication
R_9	a CS should be capable of performing cascading queries on other trusted CS that also provide the requested functionality and provide a single merged response to the user.

Table 1. Requirements for the Mixed-type SoS

R_1 contains the basic requirement of our work, which is to propose a SoS composed by a dynamic set of CS similarly to a Virtual SoS.

R_2 is important to maintain the fundamental characteristics of SoS as defined in the literature. Basically, it states that each CS should have its own set of capabilities that are kept working even when the system is not part of the SoS. Also, the system management is done independently from the fact that the system is occasionally part of the SoS.

R_3 ensures that CS are free to enter or to exit the SoS according to their own rules, i.e., a CS can manifest its interest in making part of the SoS, but can also decide to leave if this is the most appropriate action in a certain moment.

R_4 guarantees that only trustworthy systems are allowed to enter the SoS. A mechanism to catalog these systems and check if a system belongs to this catalog when trying to be part of a SoS should be provided.

R_5 depicts the duties of CS in relation to the SoS, i.e., as part of the SoS a CS can provide services that can be useful to other systems or to the SoS as whole, contributing to the emergent behavior. Eventually, a CS could be only a consumer of services provided by other CS.

R_6 is related to the dynamic configuration of CS. It is essential that new systems can be included as members of the SoS at runtime. Analogously, any CS can be removed from the list at anytime.

R_7 is important to ease the retrieval of objects throughout the SoS. There should be a mechanism through which we can identify, based only on an object unique identifier, to what CS it belongs, the identification of the CS it belongs to and its type (class).

R_8 is related to access control. Authentication, i.e. ensuring that the logged users are who they claim to be, could be managed separately by an independent CS or service. However, a CS is responsible for knowing its managed objects and mapping them to groups of users who have read/write access rights. The authorization to perform specific functions or to have access to managed objects could be dealt with by a CS or service that receives the managed objects list as input. In this case, a token could be provided as a way of ensuring that the user has been authenticated/authorized to access the system within a predetermined time period.

Finally, R_9 is related to the ability of CS to perform cascading queries. A single CS should provide the option to the user to perform cascading queries for a single query call. In this manner, the called CS needs to know other trusted CS that also provide the requested functionality of the query, which should be then called recursively, accumulating responses of the requested query into a single response to the user.

3.2. MV-SoSA Overview

Based on the specified requirements, we propose an architecture, named MV-SoSA (Mixed Virtual-System of System Architecture), that serves as a basis when composing new Mixed-type Systems of Systems. Next, we provide more details about MV-SoSA, using two different views (a conceptual model and a layer view), as well as an explanation of how cascading queries work.

3.2.1. Conceptual Model

A conceptual model for MV-SoSA was created by considering the requirements listed in Section 3.1. The diagram for this model is shown in Figure 1. *UID*, defined following R_7 and R_8, represents the unique identifier for objects of the SoS. It is inherited by *ManagedObject*, which was defined according to R_8. *ManagedObject* is used to classify objects that are stored by CS and may be transferred among other CS.

User was defined following R_5 and represents external users of the CS. *SoS* was defined according to R_1. Since the SoS is dynamic, similarly to a virtual SoS, there would be no concrete implementation of this class, because there is no entity that owns or manages the SoS completely. The class was simply created in order to represent that the SoS is composed by

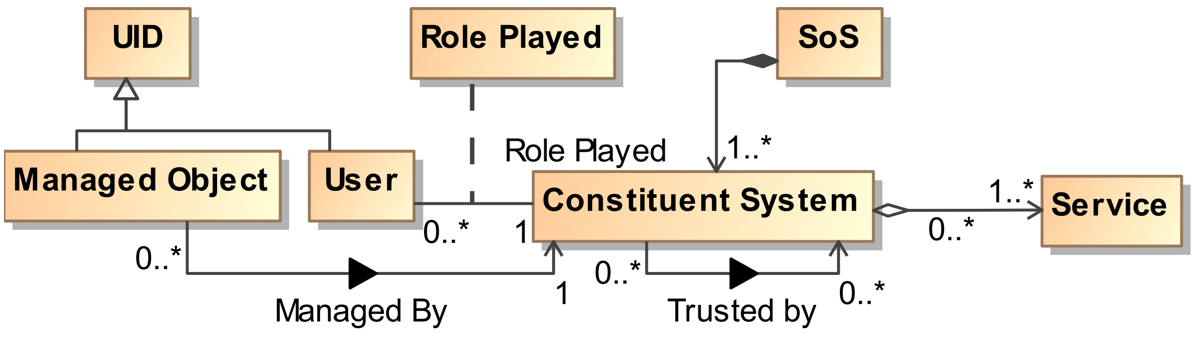


Figure 1. MV-SoSA Conceptual Model.

CS, which, in turn, is represented by *ConstituentSystem*, defined according to R_2, R_3, R_4, R_5, R_6 and R_8. Services are represented by class *Service* (R_5). These services are provided by the CS itself, to its external users or to other trusted CS.

3.2.2. Layer View

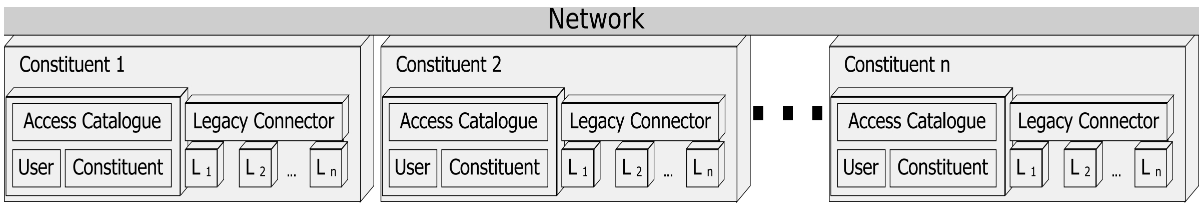


Figure 2. MV-SoSA Layer view.

The layers of the MV-SoSA architecture are shown in Figure 2. On the top of the figure, the network infrastructure is represented as a large single bus. The networking infrastructure itself is not the main objective of this architecture, it just needs to be the same throughout every CS. This way, it could be replaced by any other mechanism that enables communication of the CS.

From the perspective of external users, the CS is a single black box that provides a user profile and permission catalog, as well as other features provided as services. This happens because every call is carried out by a proxy that delegates the call to internal service providers. This proxy is also capable of calling external trusted CS in order to answer cascading queries.

The internal service providers are divided into “Access Catalogue” and “Legacy Connector”. “Access Catalogue” contains the records of trusted external entities, which are either users (including their different roles) or other CS that are entitled to access this CS. The “Legacy Connector” layer aggregates connectors which wrap the actual services targeted at the end-users of the CS. This aggregation strategy allows dynamic connector loading. Then, each CS has a variable set of features implemented by legacy service connectors, which are represented by $\mathrm{L}_{1..N}$. Each of these connectors should be implemented to provide a compatible interface to the SoS CS to be able to call the legacy service.

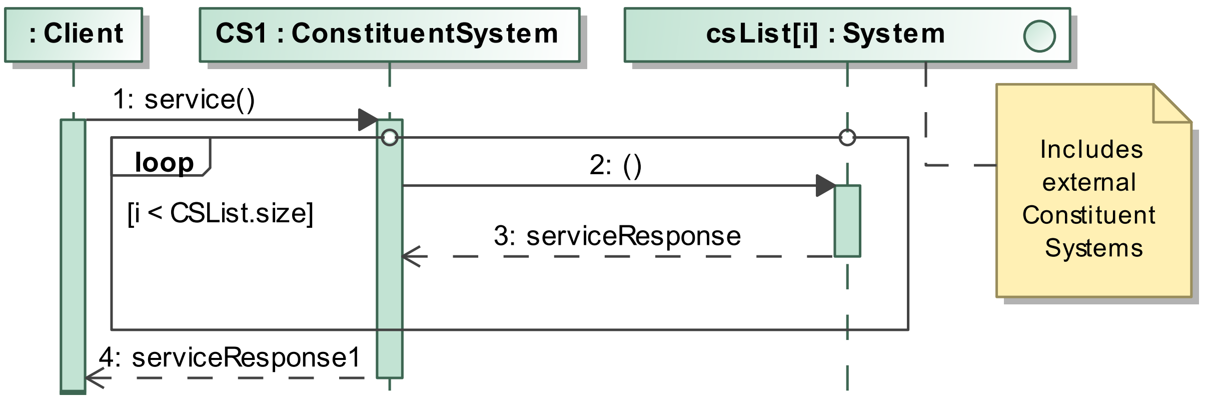


Figure 3. Sequence Diagram of Proxy Call Delegation to System Interface

3.2.3. Proxy Behavior

In order to explain the proxy behavior required to enable cascading queries, as well as dynamic feature emergence, a simplistic perspective is considered, as illustrated in Figure 3.

Consider that the Mixed-Type SoS has been successfully composed after the CS are initialized with a set of trustworthy CS. In this case, the CS ("CS1") may forward any call from external users (referred as "Client") to an internal connector capable of delegating to the requested service provider ("service"), as well as forwarding the call to any other trusted CS. The CS are actually capable of calling more than one internal connector, as well as more than one trusted CS.

The proxy behavior is similar to the proxy pattern by the Gang of Four [Gamma et al., 1995], however, in this case, the systems are dynamic. Therefore, the proxy code does not have the method signature to be called, requiring the dynamic decoding of each client request by interpreting the signature at run-time and then delegate the call to the actual legacy system and/or trusted CS. After the delegated legacy systems and trusted CS reply the request, the proxy must then aggregate the responses into a single message that is returned to the client.

4. Prototype Specification

In this section we present a prototype that follows all the requirements of MV-SoSA, referred in this paper as "MV-SoS-Prototype". This prototype is shown to demonstrate that the mixed-type is possible to be implemented, though this prototype might not cover specific kinds of SoS that are designed to integrate legacy systems. For legacy system integration, it would still require to implement connectors to attach these systems to the new architecture.

In our example, the architecture was designed as an SoS system since its inception, where every CS has basic behavior for integrating services and communicating with other CS that belong to a list of trustworthy systems. In this manner, each CS is modular and is composed by legacy connectors that provide variable sets of services.

In order to encapsulate the common behavior among CS, in our particular implementation, we consider that all CS have a kernel component. This kernel is the basic element of a CS and manages the associated services and CS administrators. The basic responsibility of the kernel is to allow communication among CS. This communication could happen among the service providers that represent actual applications for the end users or other CS. The kernel also acts as a proxy to allow external clients to access both its managed service modules, as well as with compatible services provided via other CS kernels, which boasts the emergent behavior capability.

4.1. MV-SoS-Prototype Kernel Architecture

The implemented prototype was based on a micro-kernel design with reliability as a primary non-functional feature [Tanenbaum and Bos, 2015]. An overview of the prototype class diagram is shown in Figure 4, where there are two main packages: *common* and *kernel*.

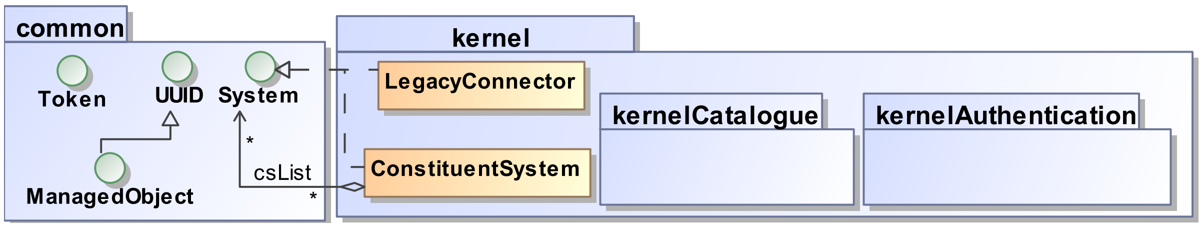


Figure 4. MV-SoS-Prototype Design

The *common* package is composed by interfaces that define data-types used throughout the architecture elements, hence the name *common*. This package is required by every element of MV-SoS-Prototype, including the kernel itself. The *kernel* package represents the actual kernel of the prototype. It was designed to cope with the most minimal and fundamental features of the system.

4.2. Common Types

Since MV-SoS-Prototype allows different CS to manage their objects, it was important to define a Universally Unique Identifier (UUID). This identifier must be hierarchical, allowing each CS to generate valid identifiers without requiring calls to other CS, which can be completely oblivious of object construction.

These identifiers are specified using the *UUID* interface, which defines how the hierarchical parts of the *UUID* are separated, as well as the minimal parts to identify a running instance of kernel and CS.

The CS based on MV-SoS-Prototype also manage objects that are addressed by these UUIDs. Therefore, the interface *ManagedObject* is employed to specify whether an object is managed, transferred and persisted by a specific CS or legacy system.

Since we have identified that the SoS requirements involve users, which could be a human user or another computer system, the *User* interface has been also defined within the *common* package. Our architecture was also planned to allow unified, yet distributed registration for users. Therefore, *User* is also treated as an universal identified object stored into the CS and shared throughout the SoS.

The *System* is an interface to define the signature of required methods needed to implement a CS based on MV-SoSA. Finally, *Token* defines how access tokens that are transferred among CS should be implemented.

4.3. Constituent Kernel

In our architecture, the Kernel of a Constituent System is designed within the *kernel* package shown in Figure 4. The kernel is the basic element of a MV-SoS-Prototype instance, which was inspired by a micro-kernel design. Therefore, it contains the execution entry point that coordinates services required to implement its functionality.

The execution entry point of the CS is defined within *ConstituentSystem* interface, which also extends the *System* of the common package. The actual implementation of the main method is defined in the *ConstituentSystem* class.

The authentication behavior is designed within the *kernelAuthentication* package. Since the kernel has an architecture inspired by micro-kernel design, authentication is defined as an external service. which also increases modularity capability. In this manner, the actual user data is not stored inside the kernel, which can delegate to more than one trusted authorization service to identify users.

Following the design of *authentication* package, the catalog is implemented as a service within *kernelCatalogue* package. Since kernels and legacy connectors are all defined as extensions of *System*, it is possible to store the legacy connectors managed by the current kernel, as well as defining trusted foreign kernels. By this established design, it is also possible to define components that serve more than one CS, increasing flexibility.

5. Evaluation - SoS Application

MV-SoSA has been applied to build a SoS for integrating instances of a reuse tool we have developed at ICMC. In this paper, this tool is simply referenced as RT. It adopts the Reusable Assets Specification (RAS) [OMG, 2005], from OMG, to store the assets in a repository, so any type of reusable asset is allowed, as for example requirements, analysis models, design models, source code, test cases, and processes. It is also based on SOA, i.e., clients can consume the provided services to manage reusable assets, which can be stored in a cloud.

When we consider several independent instances of RT running at different organizations, it is clear that each instance is benefiting from the reuse of its locally stored assets. The idea is to integrate these instances by building a SoS to offer emergent behavior that could not be obtained by independent instances. Notice that the other SoS characteristics described in Section 2, are also present in this SoS, i.e., each instance has its own management and operation; they are distributed geographically; and they could be evolved separately by different organizations, which could adapt the instance to their particular needs.

RT shares the prototype kernel that we have presented in Section 4. Therefore, RT CS can be also CS of our MV-SoS-Prototype. In order to instantiate RT it was necessary to create service connectors based on the "System" interface and to publish the service-oriented interface definitions. Then, the new CS is added to lists of other CS (part of "csList" of Figure 4). It is required to have two or more CS that provide RT Services. According to the current state of our implementation, the CS administrators must set manually the list of trusted systems. Following that possibility, if these CS trust each other, users using any of the CS are then able to access data from every instance, allowing these users to retrieve a single merged result for a query performed on any of these CS, which would not be straightforward if each CS is accessed independently. Other types of emergent behavior can be obtained as well, if necessary. For example, if the code of an asset is stored in a configuration management system, another possible emergent behavior of RT would be to warn other CS about changes and conflicts to asset users.

6. Related Work

Kazman, *et al.* have described a set of design patterns for SoS. They have described patterns such as Facade and Broker, which are similar to the Proxy Pattern, however, they have not shown this pattern itself [Kazman et al., 2013]. In this paper, we have presented an actual implementation of Proxy Design Pattern for a SoS.

Mahmood and Montagna have devised a generic framework for defining a SoS architecture suited for specific applications. They have also presented a Production SoS design by using the proposed framework [Mahmood and Montagna, 2012]. In this paper, our goal was to present a Mixed-type SoS architecture and implementation instead.

Among the definition of complex multi-tier systems that involve the definition of connected subsystems, there is the implementation of a complex robotics collaborative and distributed system [Black and Fletcher, 2006] and a transportation distributed system [Padmadas et al., 2010]. These authors describe several advantages linked to the modularity of the components employed into the design of these systems.

Vierhauser, *et al.* have proposed a software system to monitor the execution of SoS [Vierhauser et al., 2014]. In our SoS implementation, this behavior is also carried by the CS kernel, which monitors the provided subsystem services as well as trusted services provided by other CS.

Ramos devised an approach for SoS development based on statecharts and publish-subscribe mechanisms [Ramos, 2014]. His work was also based on Virtual SoS taxonomy, however, unlike our proposal, it is only suited to local area networks because it depends on event broadcast that is not suitable for a wide area network, e.g. the Internet.

7. Conclusions and Future Work

In this paper, we have established a type of SoS that fills the gap between virtual and collaborative SoS, which we referred simply as Mixed-type SoS. A set of requirements for it was presented, which characterizes a foundation for future SoS research. These requirements were used as basis to design an architecture for Mixed-type SoS, named MV-SoSA. It was designed to be simple and modular to be adapted to several application domains. To the best of our knowledge, Virtual SoS research lack exploratory studies.

Subsequently, MV-SoSA was instantiated to create an actual functional prototype, employed during an exploratory study for an application in the context of software reuse. Our implementation was successfully applied to compose independent systems fulfilling the expected requirements, e.g. emergent behavior, trusting mechanism, authentication, dynamic component loading and modularity.

We expect that our work contributes to efforts on evidencing what could or not be planned for the design of an actual Virtual SoS. Additionally, it could serve as basis to exploratory studies that can try to solve the several challenges presented by Virtual SoS.

As future work, we intend to instantiate MV-SoSA to integrate a broader range of applications pertaining to different domains, as well as conducting experiments on their usability, ease of integration and security. Among these applications, we can cite the evolution of the tool described within Section 5 to be integrated into a project for reuse support [Braga et al., 2016].

8. Acknowledgments

Our thanks to Fundação de Amparo a Pesquisa do Estado de São Paulo (FAPESP, process number 2016/05129-0) and to Coordenaçao de Aperfeiçoamento de Pessoal de Nivel Superior (CAPES) for financial support.

References

Black, R. and Fletcher, M. (2006). Simplified robotics avionics system: A integrated modular architecture applied across a group of robotic elements. In *25th Digital Avionics Systems Conference, 2006 IEEE/AIAA*, pages 1–12.

Boardman, J. and Sauser, B. (2006). System of systems - the meaning of of. In *IEEE/SMC International Conference on System of Systems Engineering*, page 6 pp.

Braga, R. T. V., Feloni, D., Pacini, K., Filho, D. S., and Gottardi, T. (2016). *AIRES: An Architecture to Improve Software Reuse*, pages 231–246. LNCS–9679. Springer International Publishing, Cham.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Kazman, R., Schmid, K., Nielsen, C., and Klein, J. (2013). Understanding patterns for system of systems integration. In *8th International Conference on System of Systems Engineering (SoSE)*, pages 141–146.

Madni, A. M. and Sievers, M. (2014). System of systems integration: Key considerations and challenges. *Systems Engineering*, 17(3):330–347.

Mahmood, A. and Montagna, F. (2012). System of systems architecture framework (SoSAF) for production industries. In *7th International Conference on System of Systems Engineering (SoSE)*, pages 543–548.

Mahmoud, H. (2005). Service-oriented architecture (SOA) and web services: The road to enterprise application integration (EAI).

Maier, M. (1998). *Architecting Principles for "Systems-of-Systems", Systems Engineering*.

Nielsen, C., Larsen, P., Woodcock, J., and Peleska, J. (2015). Systems of systems engineering: Basic concepts, model-based techniques, and research directions. *ACM Computing Surveys*, 48(2):18:1–18:41.

OMG (2005). Reusable asset specification. Available at `http://www.omg.org/spec/RAS/2.2/`.

Padmadas, M., Nallaperumal, K., Mualidharan, V., and Ravikumar, P. (2010). A deployable architecture of intelligent transportation system – a developing country perspective. In *IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, pages 1–6.

Ramos, M. A. (2014). *Bridging software engineering gaps towards system of systems development*. PhD thesis, University of Sao Paulo, Brazil.

Tanenbaum, A. and Bos, H. (2015). *Modern Operating Systems*. GOAL Series. Pearson Prentice Hall.

Vargas, I. G., Gottardi, T., and Braga, R. T. V. (2016). Approaches for integration in system of systems: A systematic review. In *Proceedings of the 4th International Workshop on Software Engineering for Systems-of-Systems*, SESoS '16, pages 32–38, New York, NY, USA. ACM.

Vierhauser, M., Rabiser, R., Grünbacher, P., Danner, C., Wallner, S., and Zeisel, H. (2014). A flexible framework for runtime monitoring of system-of-systems architectures. In *IEEE/IFIP Conference on Software Architecture*, WICSA '14, pages 57–66, Washington, DC, USA. IEEE Computer Society.

Supporting Simulation of Systems-of-Systems Software Architectures by a Model-Driven Derivation of a Stimulus Generator

Valdemar Vicente Graciano Neto[1,2,3], **Carlos Eduardo Barros Paes**[2,3], **Flavio Oquendo**[1], **Elisa Yumi Nakagawa**[3]

[1]Instituto de Informática
Universidade Federal de Goiás (UFG)
Goiânia – GO – Brazil

[2]Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP) – São Carlos – SP – Brazil

[3]Departamento de Computação – Pontifícia Universidade Católica de São Paulo (PUC-SP) – São Paulo – SP – Brazil

[4]IRISA-UMR CNRS/Université de Bretagne-Sud – Vannes – France

valdemarneto@inf.ufg.br, carlosp@pucsp.br

flavio.oquendo@irisa.fr, elisa@icmc.usp.br

Abstract. *Systems-of-Systems (SoS) often support critical domains such as air traffic control, and emergency and crisis response. Hence, their software architectures must be validated, guaranteeing that they conform to their specification. However, SoS exhibit dynamic properties that bring difficulties to a static validation. In this direction, simulations can aid by offering a dynamic view for software architecture specifications of SoS. However, to be reliable, simulation models must reproduce the real conditions of the surrounding environment in which it will be deployed. Conversely, manually stimulating simulations is tiring, repetitive, and error-prone. Then, we explored the possibility of automatically deriving a 'stimulus generator', that continuously emits stimulus for the entities being simulated, supporting an early identification of failures, and enabling correction before the deployment. In this direction, we present our method to automatically derive a stimulus generator from a software architecture description of SoS. The main contribution of this paper is providing such method, suppressing the necessity of manually coding it. We evaluated our proposal with an example of a real Flood Monitoring SoS. Preliminary results point out that the stimulus generator automatically produced is reliable, emitting the expected outputs, and suitably triggering the simulation.*

1. Introduction

Software systems have become increasingly complex, forming alliances known as Systems-of-Systems (SoS)[1], and supporting critical domains. Due to that, SoS has a

[1]For sake of simplicity, SoS will be used interchangeably to express singular and plural.

potential to cause damages, losses, and hazards. They must be constructed to be trustworthy, i.e., their operation must reliable in a way that people can safely rely and trust on their success to accomplish their missions correctly, without failures, not causing accidents, working as expected, and rearranging themselves to keep their execution [Oquendo and Legay 2015, Graciano Neto et al. 2016]. Then, it is imperative supporting the SoS software development life cycle, in particular Verification and Validation (V&V) activities, guaranteeing that the SoS yield the results expected by the missions assigned to them, with an early identification of problems in SoS operation, assuring that the SoS performs exactly what it is intended to [Michael et al. 2009].

Simulations are a recurrent approach in SoS Engineering (SoSE) to support such early anticipation of failures [Himmelspach and Uhrmacher 2007, Michael et al. 2009, Sauser et al. 2010, Zeigler et al. 2012, Mittal and Rainey 2015, Wachholder and Stary 2015]. In particular, simulations are useful for SoS, since they can externalize the inherent dynamics associated to SoS operation. However, to be reliable, SoS simulations must reproduce the real conditions in which it will be deployed. A manual approach to reproduce such stimulus, which consists of stimulating the simulation with inputs during its run-time, is costly, error-prone, tedious, and not correspondent to the real rhythm in which the stimulus could be received. Additionally, manually coding a stimulus generator is repetitive and domain-dependent, since each new SoS requires a distinct stimulus generator. In order to reduce the costs associated to its engineering, a possibility is automating the creation of such stimulus generator by a model-driven derivation based on the SoS architectural descriptions that inherently stores information about the inputs and outputs expected by the SoS.

In this paper we present specifically how to automatically derive a stimulus generator for simulation purposes of SoS software architectures by means of a model transformation. We document our SoS architecture in a high-level of abstraction using SosADL, a new architectural description language (ADL) specially tailored for SoS [Oquendo 2016]. Furthermore, we adopt DEVS (a standard formalism for simulation approaches in SoSE community) as the dynamic viewpoint to simulate SoS. Indeed, we establish a transformation from SosADL to DEVS (SosADL2DEVS) to deal with simulations of software architectures of SoS while preserving their specification. The main contribution of this paper is presenting such method, externalizing how to proceed with this transformation, extracting such information about the stimulus from an architectural description of SoS to support the creation of stimulus generators. We evaluated our proposal with a small-scale instance of a Flood Monitoring SoS. Preliminary results are promising, with an effective generation of a functional simulation. Our method brings straightforward derivation of simulation code and traceability between software architecture and simulation models. This paper is structured as follows: Section 2 outlines the concepts involved in our proposal. Section 3 distills our method. Section 4 presents a brief evaluation we carried out. Section 5 brings final considerations and points out for forthcoming work.

2. Simulation of SoS Software Architectures

Simulations are a recognized approach to deal with SoS dynamicity. They correspond to an imitation of the operation of a real-world process or system over time, and involve the generation of artificial stimulus and the observation of the effects to draw inferences concerning the operational characteristics of the real system that is represented [Banks 1999].

They (i) provide a visual and dynamic viewpoint for SoS software architectures, reproducing stimulus the system can receive from a real environment, (ii) enable the prediction of errors, diagnosing them and permitting corrections, and (iii) support the observation of expected and unexpected emergent behaviors of an SoS. In fact, simulations are one of the main groups of evaluation approaches for software architectures, typically relying on a high level implementation of the software architecture for evaluating their performance and accuracy [Bosch 2000, Santos et al. 2014]. SoS exhibit a further degree of complexity due to their dynamic nature, i.e., their operation, in particular emergent behaviors, are not visible in static specifications such as conventional software architecture documentations that adopts diagrams in UML or SySML. Thus, SoS software architectures demand an additional view that captures dynamic aspects of SoS operation.

Software architectures are often described through modern Architectural Description Languages (ADL) that offer canonical constructs to properly specify those architectures. Their formalism levels include informal (usually based on lines, rectangles, and figures denoting structures), semi-formal, and formal [Garlan et al. 2010]. Remarkable examples include Darwin (semi-formal) [Foster et al. 2011], Wright (formal) [Allen and Garlan 1997], π-ADL (formal) [Oquendo 2004], UML[2] (semi-formal) and SySML[3] (semi-formal). However, those ADL have not been developed for properly capturing SoS' dynamics [Guessi et al. 2015]. Recently, a novel ADL called SosADL was proposed for describing the architecture of SoS. It provides architectural concepts and notation formally defined in terms of the π-calculus, and also supports the specification of emergent behaviors [Oquendo 2016]. However, SosADL is not executable yet, and a dynamic approach is still required to support a plain visualization of SoS operation. Hence, an approach to support dynamic aspects of SoS software architectures descriptions is required since a long time and it is still an issue [Zave 1993, Sauser et al. 2010, Graciano Neto and Nakagawa 2015].

In this direction, simulations can offer such dynamic approach for visualization of SoS operation. Nevertheless, simulations often depend on some internal structure that imitates the surrounding environment of an SoS, delivering the stimulus that are supposed to be received by the SoS to trigger its operation [INCOSE 2016]. A stimulus generator is a virtual simulation entity responsible for playing the role of environment, such as temperature, wind, water level, and noise, or an entity which produces internal events in the systems, i.e., it imitates the reception of inputs that the constituent system can produce to itself, such as the collecting of an external data, the level of battery, of its geographic position. Developing such structure usually relies on (i) an specification of ports, inputs, outputs, and state diagram formalism in a low level of abstraction, (ii) a distinct stimulus generator for every different SoS that we produce, and (iii) a careful investigation on SoS requirements and architecture specification to elicit which stimulus should be provided, which can be costly, and error-prone. In fact, the development of stimulus generators for simulation purposes is not a new trend [Al-Hashimi 1995, Kitchen and Kuehlmann 2007]. Meanwhile, such approaches for automatically creating stimulus generator for simulation of SoS software architectures have not emerged.

Recent studies have investigated the adoption of simulation in software engineer-

[2]http://www.omg.org/spec/UML/2.5/
[3]http://www.omg.org/spec/SysML/1.4/

ing [de França and Travassos 2015, de França and Travassos 2016], and simulation has certainly been applied for SoS development [Xia et al. 2013, Graciano Neto et al. 2014, Bogado et al. 2014]. Regarding simulation of software architectures, Palladio[4] offers a solution [Becker et al. 2009]. However, there is no support for simulation of SoS software architectures. Discrete Event Systems Specification (DEVS) is a well-established formalism for simulating SoS in virtual environments [Zeigler et al. 2012]. However, it does not preserve the architectural details of SoS software architecture specification and rely in a low-level of abstraction formalism. Bogado et al. also rely on discrete event formalism for representing software architectures of monolithic software [Bogado et al. 2014], whilst Alexander et al. also deal with simulation of software architecture and dynamic aspects, but they do not address software architectures of SoS in a strict way, relying on the broader discussion of systems architecture perspective [Alexander et al. 2015].

2.1. SosADL and DEVS

In short, SosADL describes SoS, which can be expressed as a combination of architecture declarations, systems declarations, and mediators declarations[5]. An architecture declaration has an intrinsic behavior declaration, data types, and gates declarations. Gates are abstractions that enable the establishment of connections. A connection can be established to receive stimulus from or act on the environment, or simply for a communication between constituents. Furthermore, the connection can be for input, output, or for both. Data types can have inherent functions, and functions can have expressions associated. Mediators and systems, as well as the SoS architecture itself, have gates, data types, and behaviors. Systems play the role of constituents in an Architecture Behavior Declaration, and Systems are mediated by mediators. SosADL supports emergent behavior representation by the idea of coalition, a temporary alliance for combined action among constituents connected by mediators. Those behaviors are specified as part of the coalition behavior, documenting how constituents should interact to accomplish a given set of missions[6].

In turn, DEVS is structured based on atomic and coupled models. Atomic models represent constituents, and coupled models represent the communications among constituents, materializing the SoS as a whole and constituents' interoperability. In DEVS, Atomic models have the following elements: (i) a labeled state diagram, that performs transitions due to input or output events; (ii) abstract data types definition, (iii) global variables definition, (iv) variables initialization, (v) ports definition, and (vi) events definition. An atomic model with only a state diagram specification and ports definition is already executable. Coupled models are expressed as a System Entity Structure (SES), i.e., a formal structure governed by a small number of axioms that expresses how atomic models communicate. A straightforward generation of DEVS code does not guarantee the simulation be executable. This happens because the SoS operation is deeply related to the stimulus received from the environment that triggers the accomplishment of a mission. Thus, it is necessary to elaborate a specific entity in the simulation model that is responsible for delivering the expected stimulus that drive the operation of the SoS: the stimulus generator. Next section discusses how to automatically produce such stimulus

[4]http://www.palladio-simulator.com/

[5]Mediators are architectural elements that establish communication between two or more constituents

[6]More details about the syntax of architecture descriptions in SosADL and its elements can be found in [Oquendo 2016].

generator from SosADL models, properly transforming SosADL in DEVS.

3. A Model-Driven Derivation of a Stimulus Generator for Simulation of SoS Software Architectures

We present our approach relying on a real example: a Flood Monitoring SoS (FMSoS), i.e., a SoS composed of smart sensors (i.e., complex sensors that embed software) to monitor the occurrences of floods in an urban area in the city of São Carlos in Brazil. Rivers cross the city and, when the rains are intense, floods recurrently occur, causing losses, damage, and iminent danger for population. The FMS we describe is concerned to validate a single emergent behavior: *flood alert*. Such SoS consists of a collaborative SoS, in which there is not a central authority that orchestrate the constituents' functionalities to accomplish missions. Figure 1 gives an illustration of the FMSoS[7]. Sensors are spread on the river's edges with a regular distance among them, and mediators exist between every pair of sensors in a pre-established distance between them. The data collected by sensors are collected and transmitted until reaching the gateway. In case of flood, the gateway emit an alarm for the public authorities. All of the codes presented henceforth are based on this example.

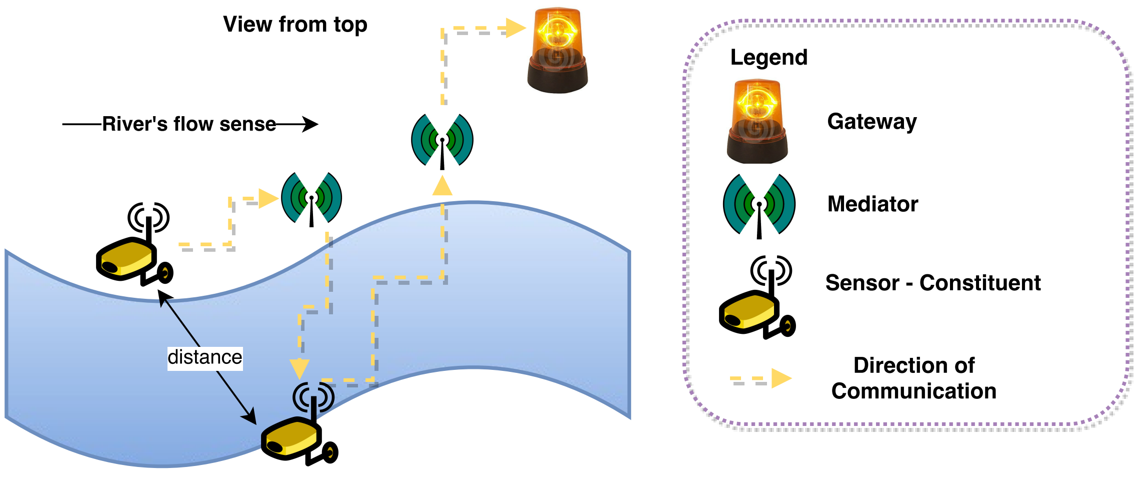


Figure 1. A Flood Monitoring System-of-System (FMSoS).

FMSoS is an SoS, since it exhibits [Maier 1998]:
Operational independence of the constituents, i.e., smart sensors perform their own missions despite being out of the scope of the SoS;
Managerial independence of constituents, since a diversity of stakeholders and enterprises might independently own, deliver, and manage the smart sensors;
Distribution, since they interoperate through a communication network;
Evolution, since the SoS evolves as a consequence of changes in the configuration or functionality of smart sensors; and
Emergent behavior, since one unique constituent could not deliver a flood alert by itself. For instance, if only one sensor performs its activities in an urban area, it could not notify a flood on time, being not effective. Conversely, if it is used outside of the urban area,

[7]Credits for the images used to compose the figure: http://goo.gl/TTOlAa, http://goo.gl/QCUAKY, http://goo.gl/a9Y0Dw.

trying to previously diagnostic the flood, it would be so far that could not reach a gateway to communicate to communicate the danger. Also, it might emit a false alert, since the flood could be limited to another place. Hence, the flood alert is a result of the interoperability among a diversity of constituents working in cooperation, spread along the river edge. Furthermore, we deal with a weak emergent behavior, since they can not be predicted by static models (simple emergent behaviors), and they are reproducible in simulations (but not in static models), emerging predicted behaviors but still with potential to non-predicted behaviors that must be handled.

We selected SosADL as the SoS software architecture specification notation, and DEVS as the simulation formalism. We adopted a DEVS dialect called DEVS Natural Language (DEVSNL) that enables programming atomic and coupled models in a human-like format in tools such as MS4ME[8]. A SosADL code is submitted as input for a XTend script that materializes the Code Generator. A functional code written in DEVSNL is generated as output, establishing a SosADL2DEVS transformation. Briefly, considering a broad view of the transformation, the concepts of System and Mediator in SosADL are transformed into Atomic Models in DEVS. Behaviors are materialized into labeled state diagrams in DEVS that configures the constituents operation. Architecture, Coalition, and SoS become Coupled Models. Connections and gates become Ports, and Data Types and Functions are mapped in data types and functions in DEVS.

Each statement (line of code) within a SosADL behavior is converted in one or more state transitions in DEVS. In DEVS, transitions can occur due to (i) a data received, expressed as `?data`, (ii) a data sent, expressed as `!data`, and (iii) a spontaneous transition, without any input or output. This is the approach we used to generate constituents, mediators, and gateway[9]. However, the derivation of the stimulus generator is quite different. In SosADL, there is a special type of connection called **environment**, that abstracts interaction of an SoS with the surrounding environment, emitting outputs to the environment, or receiving stimulus from it, e.g., when the system is a sensor. Moreover, SosADL offers a library called Localization, that offers invoking Global Positioning System (GPS) functionalities.

All of the SosADL aspects must be traduced to DEVSNL to create a functional simulation. However, there are no straightforward elements in DEVSNL to automatically produce environment stimulus. Thus, it is necessary to create a stimulus generator that delivers the expected inputs the constituents wait to perform transitions and to start their behavior execution. SosADL models are analyzed by the transformation algorithm, searching for environment connections and callings for localization libraries. For convention, constituents are analyzed first, since they are the frontier of a SoS, and state transitions for the stimulus generator are created to serve their necessities. After that, Mediators are investigated. Lastly, Gateways are analyzed. Since usually constituents starts the SoS operation due to the stimulus they receive from environment, we prioritize the analyzes of constituents. If there are congruent statements between constituents and mediators, i.e., a same type of stimulus that must be received for both, a unique transition is created in the stimulus generator. Each connection specified as an `environment` con-

[8]http://goo.gl/NmBBuu

[9]We do not discuss this mechanism with details in this paper, since the focus is the representation and derivation of a stimulus generator. Other details are discussed in a forthcoming paper.

nection produces one transition in the specification of the state diagram in the resulting stimulus generator. Hence, the stimulus generator consists of a special type of system (in the context of the simulation) that has a continuous behavior (a behavior materialized as a loop) to emit stimulus by output state transitions, starting and keeping the SoS in operation.

In DEVS, such stimulus generator is also represented as an Atomic Model. Listing 1 shows an excerpt of a code in SosADL that specifies one of the sensors that compose the FMSoS. Some parts are hidden since they do not influence in the discussion of stimulus generation derivation. It is possible to see that the gate energy offers two environment connections (Lines 12 and 13): one to receive a threshold (a limit of energy that is considered enough to keep the sensor in operation), and `power`, that is used to receive the level of battery available. Within the behavior `sensing`, it is possible to see the sensor receiving its coordinate and receiving the energy threshold and power level. In turn, Listing 2 shows the code in DEVSNL that specifies the stimulus generator produced using our approach. We changed the names of the states to become them more readable. It illustrates not only the transitions generated from the code of the sensor showed in Listing 1, but also from the codes of mediators and gateway (Lines 5 to 15). In Listing 2, the stimulus generator has three output ports (Lines 1 to 3) that simulates the collection of the geographic positions (lps), power level, and the reception of the water level by the mediators, sensors, and gateway. Figure 3 depicts a state diagram equivalent to the DEVS code presented in Listing 2. It delivers the aforementioned data, and comes back to the state `LPSsent`, forming a loop that keeps the stimulus continuously running and offering the stimulus for the operation of the SoS.

4. A Brief Evaluation

We adopted a specification in SosADL of a real Flood Monitoring SoS already in operation to evaluate our proposal [Horita et al. 2015]. Our aim was evaluating if the simulation (automatically produced) would run as expected and deliver, as a result, a single emergent behavior. A brief video demonstrating the generated simulation is available externally[10]. It shows the simulation running and the stimulus generator successfully delivering the outputs necessary to the simulation execution. Indeed, the approach was submitted to domain experts. They classify our approach as suitable to generate a stimulus generator that is reliable and correspondent to the stimulus specified in high-level in the SoS software architecture specification.

Regarding threats to validity, we can mention the possibility of failures if the SoS architect forget to qualify the environment connections in SosADL with the keyword `environment`. If it occurs, the simulation can fail, since the expected input can be never received. Indeed, any error regarding the declaration of environment connections at design-time can affect the final simulation. Moreover, more accurate evaluation with larger contexts and applications are still required.

5. Final Remarks

This paper presented a model-driven solution to automatically derive a stimulus generator to be used in a simulation of SoS software architectures. Our proposal contributes

[10]`https://goo.gl/pdGCIC`

Listing 1. A specification of a Sensor in SosADL.

```
//'with' imports declarations suppressed
// Description of Sensor as a System Abstraction
library WsnSensor is {
  system Sensor( lps:Coordinate ) is {
  // Declaration of local types hidden
  gate measurement is {
      connection pass is in { MeasureData }
      connection measure is out { MeasureData }
  }

    gate energy is {
      environment connection threshold is in { Energy }
      environment connection power is in { Energy }
    }
    gate location is {
      connection coordinate is out { Coordinate }
    }

    behavior sensing is {
          via location::coordinate send sensorcoordinate
      via energy::threshold receive powerthreshold
      repeat {
        via energy::power receive powerlevel
        if( powerlevel > powerthreshold ) then {
                  choose {
            via measurement::sense receive data
            via measurement::measure send tuple {
            coordinate = lps, depth = data::convert( ) }
          } or {
            via measurement::pass receive data
            via measurement::measure send data
 } } }}}}
```

Listing 2. Code DEVSNL for a stimulus generator.

```
generates output on lps!
generates output on powerLevel!
generates output on sense!

to start hold in s0 for time 1!
after s0 output lps!
from s0 go to LPSsent!
hold in LPSsent for time 1!
after LPSsent output powerLevel!
from LPSsent go to powerLevelSent!
hold in powerLevelSent for time 1!
after powerLevelSent output sense!
from powerLevelSent go to
    waterLevelSent!
hold in waterLevelSent for time 10!
from waterLevelSent go to LPSsent!
```

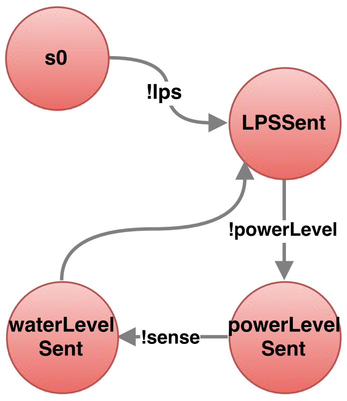


Figure 2. A State Diagram equivalent to the DEVS code generated for the stimulus generator of a FMSoS.

by automating the process of generation of that stimulus generator, bringing productivity, traceability between the models. This solution is a proposal of a joint effort of two research groups: SofTware ARchitecture Team (START/ICMC-USP) and ArchWare (IRISA/UBS), and is part of a broader approach to support validation of emergent behaviors in software architecture of SoS by the adoption of model driven derivation of a simulation of SoS. Future works include scaling the solution, testing the confidence of the model-driven transformation, and conducting experimental studies to evaluate our proposal.

Acknowledgments: We thank CNPq (grant:201230/2015-1/SWE) and FAPESP (grant: 2014/02244-7).

References

Al-Hashimi, B. (1995). *The Art of Simulation Using PSpice: Analog and Digital.* CRC Press, Inc., Boca Raton, FL, USA, 1st edition.

Alexander, P., Nicolaescu, A., and Lichter, H. (2015). Model-based evaluation and simulation of software architecture evolution. ICSEA, pages 153 – 156.

Allen, R. and Garlan, D. (1997). A formal basis for architectural connection. *ACM Trans. Softw. Eng. Methodol.*, 6(3):213–249.

Banks, J. (1999). Introduction to simulation. In *Proceedings of the 31st Conference on Winter Simulation: Simulation—a Bridge to the Future - Volume 1*, WSC '99, pages 7–13, New York, NY, USA. ACM.

Becker, S., Koziolek, H., and Reussner, R. (2009). The palladio component model for model-driven performance prediction. *J. Syst. Softw.*, 82(1):3–22.

Bogado, V., Gonnet, S., and Leone, H. (2014). Modeling and simulation of software architecture in discrete event system specification for quality evaluation. *Simulation*, 90(3):290–319.

Bosch, J. (2000). *Design and Use of Software Architectures: Adopting and Evolving a Product-line Approach.* Addison-Wesley, New York, USA.

de França, B. B. N. and Travassos, G. H. (2015). Simulation based studies in software engineering: A matter of validity. *CLEI Electron. J.*, 18(1).

de França, B. B. N. and Travassos, G. H. (2016). Experimentation with dynamic simulation models in software engineering: planning and reporting guidelines. *Empirical Software Engineering*, 21(3):1302–1345.

Foster, H., Mukhija, A., Rosenblum, D. S., and Uchitel, S. (2011). *Rigorous Software Engineering for Service-Oriented Systems: Results of the SENSORIA Project on Software Engineering for Service-Oriented Computing*, chapter Specification and Analysis of Dynamically-Reconfigurable Service Architectures, pages 428–446. Springer, Berlin, Heidelberg.

Garlan, D., Bachmann, F., Ivers, J., Stafford, J., Bass, L., Clements, P., and Merson, P. (2010). *Documenting Software Architectures: Views and Beyond.* Addison-Wesley Professional, 2nd edition.

Graciano Neto, V. V., Guessi, M., Oliveira, L. B. R., Oquendo, F., and Nakagawa, E. Y. (2014). Investigating the model-driven development for systems-of-systems. ECSAW '14, pages 22:1–22:8, New York, NY, USA. ACM.

Graciano Neto, V. V. and Nakagawa, E. Y. (2015). A biological inspiration to support emergent behavior in systems-of-systems development. WDES' 15, pages 33–40, Belo Horizonte, Brazil. SBC.

Graciano Neto, V. V., Oquendo, F., and Nakagawa, E. Y. (2016). Systems-of-systems: Challenges for information systems research in the next 10 years. GRANDSI-BR/SBSI, pages 1–3, Florianópolis, Brazil. SBC.

Guessi, M., Graciano Neto, V. V., Bianchi, T., Felizardo, K. R., Oquendo, F., and Nakagawa, E. Y. (2015). A systematic literature review on the description of software architectures for systems of systems. In *SAC*, pages 1433–1440, Salamanca, Spain.

Himmelspach, J. and Uhrmacher, A. M. (2007). Plug'n simulate. ANSS '07, pages 137–143, Washington, DC, USA. IEEE Computer Society.

Horita, F. E., de Albuquerque, J. P., Degrossi, L. C., Mendiondo, E. M., and Ueyama, J. (2015). Development of a spatial decision support system for flood risk management in Brazil that combines volunteered geographic information with wireless sensor networks. *Computers & Geosciences*, 80:84 – 94.

INCOSE (2016). The Guide to the Systems Engineering Body of Knowledge (SEBoK). SEBoK v. 1.6 released 25 March 2016.

Kitchen, N. and Kuehlmann, A. (2007). Stimulus generation for constrained random simulation. ICCAD '07, pages 258–265, San Jose, California. IEEE Press.

Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering*, 1(4):267–284.

Michael, J. B., Riehle, R., and Shing, M. T. (2009). The verification and validation of software architecture for systems of systems. In *SoSE*, pages 1–6, Albuquerque, USA.

Mittal, S. and Rainey, L. (2015). Harnessing Emergence: The Control and Design of Emergent Behavior in System of Systems Engineering. In *SCS*, pages 1–10, San Diego, CA, USA. SCSI.

Oquendo, F. (2004). π-ADL: An Architecture Description Language Based on the Higher-order Typed π-calculus for Specifying Dynamic and Mobile Software Architectures. *SIGSOFT Softw. Eng. Notes*, 29(3):1–14.

Oquendo, F. (2016). Formally Describing the Software Architecture of Systems-of-Systems with SosADL. In *SOSE*, pages 1–6, Kongsberg, Norway. IEEE.

Oquendo, F. and Legay, A. (2015). Formal Architecture Description of Trustworthy Systems-of-Systems with SosADL. *ERCIM News*, (102).

Santos, D. S., Oliveira, B., , 1, M. G., Oquendo, F., Delamaro, M., and Nakagawa, E. Y. (2014). Towards the evaluation of system-of-systems software architectures. WDES, pages 53 – 57. SBC.

Sauser, B., Boardman, J., and Verma, D. (2010). Systomics: Toward a Biology of System of Systems. *IEEE Trans. on Systems, Man, and Cybernetics*, 40(4):803–814.

Wachholder, D. and Stary, C. (2015). Enabling emergent behavior in systems-of-systems through bigraph-based modeling. In *SOSE*, pages 334–339, San Antonio, USA. IEEE.

Xia, X., Wu, J., Liu, C., and Xu, L. (2013). A model-driven approach for evaluating system of systems. In *ICECCS*, pages 56–64.

Zave, P. (1993). Feature interactions and formal specifications in telecommunications. *Computer*, 26(8):20–29.

Zeigler, B. P., Sarjoughian, H. S., Duboz, R., and Souli, J.-C. (2012). *Guide to Modeling and Simulation of Systems of Systems*. Springer.

Distributed Coding Dojo Randori: An Overview and Research Opportunities

Bernardo Estácio[1], **Josiane Kroll**[1] **Rafael Prikladnicki**[1]

[1]PUCRS - Pontifícia Universidade Católica do Rio Grande do Sul
Computer Science School – Porto Alegre – RS – Brazil

{bernardo.estacio,josiane.kroll }@acad.pucrs.br, rafaelp@pucrs.br

Abstract. *Distributed Software Development (DSD) is widely adopted in the software industry. Its adoption require highly responsive teams that can support a product development in an environment that poses several challenges. Coding Dojo Randori (CDR) is an collaborative software practice for training people that offers the opportunity for developing new skills. Thus, CDR can be applied for different contexts that includes DSD. In this paper, we present an initial discussion about the concept, hereby called Distributed Coding Dojo Randori, by analyzing untapped research opportunities for the area.*

1. Introduction

Distributed Software Development (DSD) is a trend in the software industry. Companies are looking for benefits such as the reduction of high travel costs, the access of a geographically skilled staff as well as the possibility to innovate and share best practices inside the organization [Conchúir et al. 2009]. Despite that, efforts are still need for evaluating the applicability of coding practices in distributed contexts, specially for those dealing with agile development [Jalali and Wohlin 2010].

Coding Dojo Randori (CDR) is a collaborative practice that can be applied for different software development contexts. It is defined as a collaborative practice where a group of developers train and learn about some technology concept (programming language, framework). CDR is particularly known as a technique to train agile concepts as Test-Driven Development (TDD) [Sato et al. 2008].

CDR have been practiced in the software industry over the years [Rooksby et al. 2014], but a little is known about this practice in DSD contexts. In order to better understand the Distributed Coding Dojo Randori (DCDR), we present an initial discussion about this concept and untapped research opportunities for the area.

2. Coding Dojo

A Coding Dojo is defined as a meeting where a group of programmers gets together to learn, practice, and share experiences. There are different formats for Coding Dojo. All of them depends on the rules that participants should follow. The main formats adopted in the software industry are Kata and Randori.

In the Kata format, exercises are practiced in advance and then performed in front of the audience during the meetings [Rooksby et al. 2014]. Instead of showing the final code and tests, the presenters start from scratch, explaining each step and allowing the other participants to ask questions or make suggestions. The main goal here is making everyone to reproduce the steps and solve the same problem after the meeting [Sato et al. 2008].

The Randori format consists of a group of developers working together to solve a specific task. One participant acts as the driver, another one as the navigator, and the remaining participants are the audience that can also participate.The roles of individuals are changed in rounds. Sato et al. [Sato et al. 2008] recommends that each round should last from 5 to 7 minutes. At the end of a round, the driver moves to the audience, the navigator turns into the driver and someone of the audience starts to act as a navigator [Sato et al. 2008].

Despite Randori format is being widely practiced, in the literature there are few studies that evaluate this technique. Sato et al. [Sato et al. 2008] presents how is the different formats with some lessons learned about the organization of a Coding Dojo session in industry and academy. Da Luz et al.[Da Luz et al. 2013] reports a investigation about the learning of TDD in Randori sessions. Though the perceptions of participants, the Randori session helped them to learn TDD.

3. Distributed Coding Dojo Randori

Since 1990 when the first research report in DSD was published [Prikladnicki et al. 2010], studies have been providing an understanding about DSD challenges. Studies have discussed the importance of training teams for DSD [Damian et al. 2006]. Thus, CDR as seem as an opportunity for DSD. For the purposes of this position paper, we define "Distributed Coding Dojo Randori" as "an collaborative practice based on Coding Dojo Randori concept applied em Distributed Software Development environments". Figure 1 shows the research topic emerged from the union of DSD and CDR.

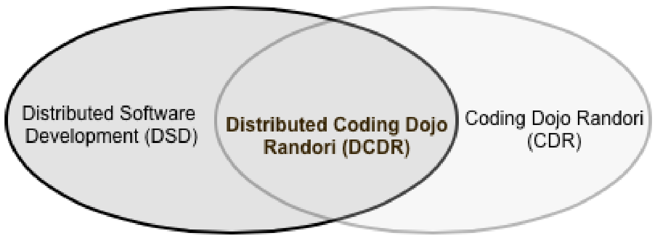

Distributed Software Development (DSD)
Distributed Coding Dojo Randori (DCDR)
Coding Dojo Randori (CDR)

Figure 1. Distributed Coding Dojo Randori research topic.

There are a number of reasons that CDR makes particular sense for DSD. First, for training distributed software teams, in particular newcomers or new hires, to introduce or practice on a particular new concept (programming language, technique). Second, CDR fits to agile software development context, for instance TDD. Finally, DCDR can help in the knowledge transfer of the project among the distributed teams.

Traditional or agile development in DSD involves team members distributed in different places, countries or even continents. Any activity within the software development life cycle (SDLC) can be executed in DSD. Teams geographically dispersed can work in testing, coding, designing or any activity. There are many team settings and examples range from remote sub-teams producing specific modules of a system to teams where different functional roles such as developer or business analysis are executed at different locations [Lane and Agerfalk 2008]. Some examples where CDR could be applied in DSD are described bellow.

Newcomers in Distributed Teams: DCDR can help new members to get introduced in the distributed team, mainly due the close collaboration perspective that practice has. Through the DCDR the sessions would be possible to promote a socialization among the team as well to support the alignment and intregration among the developers.

Agile Software Development: There is a growing interest to evaluate Agile and DSD [Jalali and Wohlin 2010]. In this context, DCDR can be applied through the team especially to train agile techniques such as TDD and Refactoring.

Knowledge Transfer: The different levels of knowledge transfer are challenging for DSD [da Silva et al. 2010]. DCDR can help the team to get in contact together in the project, spreading the knowledge among the developers. DCDR can also be used to solve impediments in the project together, distributing the knowledge between different sites and developers.

4. Research Opportunities for DCDR

There are many research opportunities for DCDR. Here we categorize these opportunities in three large research topics that can be spread in specific topics. Next, we present each of them.

Context of application: CDR can be adopted in academic and industry contexts [Sato et al. 2008]. Thus, global software education brings an opportunity to evaluate the adoption of DCDR in courses to understand how the distributed teams work combined with software tasks to train and learn a specific concept in programming, such as TDD. There are some aspects that should be investigated such as the presence of a tutor, the configuration of the courses, and also how the learning can be assessed.

With the skilled professionals distributed, DCDR can help especially in knowledge transfer between the teams. In addition, DCDR can be applied to train teams in a specific concept, technology, or just to facilitate new distributed team members to get familiar with the project and technology used. There are challenges in adopting DCDR, especially the number of teams members, the task complexity and also the level of experience between the professionals.

Tools to support DCDR: Tools are essential to make DCDR works. There are opportunities to investigate which existing tools are useful to support the practice properly. Also, we need to analyze if there are special requirements or features that need to be develop for them, and how can we develop those.

Tools should support the rules of Randori (driver, navigator, and audience) and provide screen sharing with text and audio channel. In this context, it is important to assess if the infrastructure for synchronous code-collaboration with a group of participants is feasible or not.

DSD settings and aspects: the software development distribution involves several challenges such as coordination, culture, language, and organization of the company in terms global development. These aspects need to be investigate in DCDR to identify how they can impact in the dynamic of the practice.

There is an opportunity to analyze if the rules in the Randori format should be modified or not, for instance the participation of the audience during the session. Also, the coordination of the session through the presence of a organizer and how can we can support it.

5. Final Remarks

In this paper we promote an initial discussion towards the research opportunities in DCDR. We summarize the relevant information about the topic and untapped research opportunities for the area. As next steps, we plan to run empirical studies to contextualize DCDR and analyzed its benefits and limitations of the practice. We are also planning to further identify the main variables that can influence the teams training in distribute settings.

References

Conchúir, E. O., , P. J., Olsson, H. H., and Fitzgerald, B. (2009). Global software development: Where are the benefits? *Commun. ACM*, 52(8):127–131.

Da Luz, R., Serra Seca Neto, A., and Vida Noronha, R. (2013). Teaching TDD, the coding dojo style. In *IEEE 13th International Conference on Advanced Learning Technologies (ICALT), 2013*, pages 371–375.

da Silva, F. Q. B., Costa, C., Franca, A. C. C., and Prikladinicki, R. (2010). Challenges and solutions in distributed software development project management: A systematic literature review. In *2010 5th IEEE International Conference on Global Software Engineering*, pages 87–96.

Damian, D., Hadwin, A., and Al-Ani, B. (2006). Instructional design and assessment strategies for teaching global software development: A framework. In *International Conference on Software Engineering*, pages 685–690, Shanghai, China. IEEE.

Jalali, S. and Wohlin, C. (2010). Agile practices in global software engineering - a systematic map. In *2010 5th IEEE International Conference on Global Software Engineering*, pages 45–54.

Lane, M. T. and Agerfalk, P. J. (2008). On the suitability of particular software development roles to global software development. In *2008 IEEE International Conference on Global Software Engineering*, pages 3–12.

Prikladnicki, R., Audy, J. L. N., and Shull, F. (2010). Patterns in effective distributed software development. *IEEE Softw.*, 27(2):12–15.

Rooksby, J., Hunt, J., and Wang, X. (2014). Agile processes in software engineering and extreme programming. chapter The Theory and Practice of Randori Coding Dojos, pages 251–259. Springer-Verlag, Berlin, Heidelberg.

Sato, D., Corbucci, H., and Bravo, M. (2008). Coding dojo: An environment for learning and sharing agile practices. In *Agile Conference, 2008*, pages 459–464.

Transparência em Ecossistemas de Software

Rodrigo Santos[1,2], Claudia Cappelli[1], Cristiano Maciel[3], Julio Cesar Sampaio do Prado Leite[4]

[1]DIA/UNIRIO, Universidade Federal do Estado do Rio de Janeiro
CEP 22290-240 – Rio de Janeiro, RJ, Brasil

[2]COPPE/UFRJ – Universidade Federal do Rio de Janeiro
CEP 21941-972 – Rio de Janeiro, RJ, Brasil

[3]LAVI/UFMT, Universidade Federal de Mato Grosso
CEP 78060-900 – Cuiabá, MT, Brasil

[4]DI/PUC-Rio, Pontifícia Universidade Católica do Rio de Janeiro
CEP 22451-900 – Rio de Janeiro, RJ, Brasil

{rps, claudia.cappelli}@uniriotec.br, cmaciel@ufmt.br,
www.inf.puc-rio.br/~julio

Resumo. *A diversidade de abordagens para construir sistemas alavancou plataformas globalizadas, de larga escala e de longo prazo. Tais plataformas têm sido vistas como núcleos de ecossistemas e envolvem aspectos técnicos, econômicos e sociais em sua construção e evolução. Dado que ecossistemas são formados por uma comunidade de orquestradores e colaboradores, lidar com transparência é um fator crítico. Este artigo apresenta uma discussão preliminar sobre a importância de se pensar a transparência em ecossistemas de software e identifica desafios e oportunidades para a sua implementação.*

Abstract. *The diversity of approaches to build software systems led to the emergence of globalized, large-scale, and long-term platforms. Such platforms have been seen as ecosystems cores and involve technical, economic and social aspects in their development and evolution. Since the ecosystems are formed by a community of orchestrators and niche players, the need to handle transparency is a critical factor. This paper presents an initial discussion on the importance of thinking about the transparency in software ecosystems and identifies challenges and opportunities for its implementation.*

1. Introdução

Conforme discutido por Cataldo e Herbsleb (2010), modelos de desenvolvimento de software têm lidado com o surgimento de plataformas globalizadas, de larga escala e de longo prazo. Os desafios desse cenário não mais se limitam à construção e evolução de um projeto ou produto único. Santos *et al.* (2014) afirmam que, ao reunir diversos projetos e produtos em torno de uma tecnologia de software central, essas plataformas originam sistemas mais complexos, que integram uma rede de diversos atores e artefatos, internos e externos, denominados Ecossistemas de Software (ECOS). Nesse contexto, organizações produtoras de software (orquestradoras) têm sofrido pressão do mercado para abrir plataformas e envolver desenvolvedores externos (colaboradores). Portanto, o elemento "software" passa a não existir mais de maneira "fechada", ficando distribuído e mantido pelas comunidades criadas ao seu redor (Manikas, 2016).

Baseado nos resultados de Barbosa *et al.* (2013), observa-se que tal abertura produz algumas indagações: “entre plataformas livres ou proprietárias, qual gera maior sucesso?”, “qual estratégia recebe maior atenção da comunidade de desenvolvedores?” e “qual delas produz aplicações mais inovadoras?”. Nesse contexto, a necessidade de lidar com transparência é um fator crítico para um desenvolvimento de software de qualidade (Leite e Cappelli, 2010). Um problema está em como produzir/adquirir software e gerir negócios a partir de informações “consumidas” via ECOS, de forma transparente. Este artigo apresenta uma discussão preliminar sobre a importância de se pensar a transparência em ECOS e identifica desafios e oportunidades para a sua implementação. Para isso, a Seção 2 discorre sobre transparência; a Seção 3 introduz este conceito no contexto de ECOS; a Seção 4 traz uma agenda de pesquisa; e a Seção 5 conclui o artigo.

2. Transparência

Transparência tem sido uma preocupação crítica para a sociedade moderna (Holzner e Holzner, 2006). A importância da abertura de informações sobre o funcionamento de sistemas se deve à necessidade de atores capazes de entender e acessar as informações disponíveis. Lord (2006) estabelece transparência como condição para que informações relativas a prioridades, capacidades e comportamentos estejam amplamente disponíveis. No Brasil, por meio da Lei de Acesso à Informação nº 12.527 (Brasil, 2011), foram estabelecidos vários elementos para garantir o acesso da sociedade a informações públicas. Observando essa e outras necessidades, Cappelli (2009) definiu transparência como um conjunto de características que permite fornecer aos interessados informações gerais sobre *acessibilidade*, *usabilidade*, *informativo*, *entendimento* e *auditabilidade*. A Figura 1 apresenta essas características reunidas para criar o conceito de transparência.

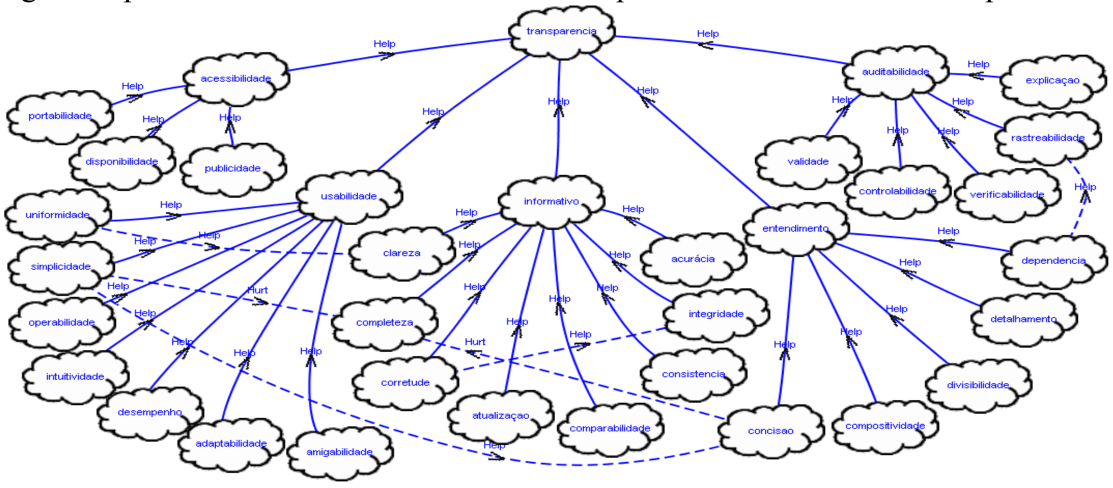


Figura 1. Grafo de interdependência de metas flexíveis em transparência (Cappelli, 2009)

Todavia, a geração de informação nas organizações e processos organizacionais são, em sua grande parte, realizados pelo uso de software, tornando-se necessária a extensão do conceito de transparência ao software e aos processos de desenvolvimento deste. Os padrões de desenvolvimento atuais têm buscado promover a disponibilização das informações em formatos abertos e acessíveis, a fim de possibilitar a reutilização e a interligação com informações de outras fontes, gerando novos significados. Isso tem sido possível com o uso do conceito de Dados Governamentais Abertos (Germano e Takaoka, 2012), que tem relação com a transparência (Cruz *et al.*, 2016). No entanto, conforme a Figura 1, isso é insuficiente, face às várias características que contribuem para a transparência. Quanto mais contribuições (*help*) receber a meta-flexível *transparência,* maior será o seu grau de satisfatibilidade (Leite e Cappelli, 2010).

3. Transparência em Ecossistemas de Software

Analisar um ECOS é mais complicado do que simplesmente separar parte técnica e não técnica de um sistema (Cukierman *et al.*, 2007), pois o ecossistema pode ter diferentes usos ou aplicações. Veja o caso, por exemplo, de uma biblioteca universitária que demanda um sistema para gestão do acervo, com integração com outras universidades, com o portal de periódicos da Capes, com grupos de pesquisa, e atendendo a diferentes tipos de usuários. Esse sistema se caracteriza por envolver diferentes softwares de terceiros, além de uma gama de diferentes atores. Portanto, é adequado tratá-lo com a ótica de ECOS, de forma a melhor compreender como os vários softwares colaboram. Tal sistema lidará com vários tipos de estímulos, notadamente controlados por humanos, que desenvolverão, operarão e/ou serão usuários das facilidades de uma biblioteca e, com isso, contarão com uma infraestrutura de hardware e software selecionada para tal. Se compararmos os diversos tipos de atores, veremos que existem diferenças em seus interesses frente ao ECOS. Por exemplo, existem interesses: a) relativos ao portal da Capes, b) de como se dará a comunicação com outras universidades ou grupos de pesquisa, c) de usuários da biblioteca dessa universidade, e d) de uso/cessão de softwares de terceiros. O fato de haver diferenças entre interesses requer a existência de uma política de transparência no ECOS.

Nesse sentido, a ausência da devida transparência do que cada interessado tem como política de convivência, isto é, como está disposto a interagir/comunicar-se, seria um grande obstáculo para a construção desse sistema. Portanto, é importante que todos que ali convivem saibam sobre os demais interesses, permitindo negociação e consenso entre os participantes. Isso envolve gerir requisitos desejados e objetivos estabelecidos para o sistema (Fotrousi *et al.*, 2014). A definição do ecossistema é, portanto, função da percepção desses atores, sejam eles orquestradores (*hubs*) ou colaboradores (*niche players*), podendo ter matizes diferentes. Pode-se pensar em ecossistemas centrados na infraestrutura como, por exemplo, em uma plataforma móvel, ou no caso de aplicações transacionais, no qual o foco está em uma tecnologia, como o sistema gerenciador de banco de dados. Pode-se ainda centrar o ecossistema no ambiente externo, no qual o foco são os atores que operam e aqueles que usam os serviços como, por exemplo, *sites* de redes sociais e ecossistemas de *startups*. Supõe-se que um maior conhecimento compartilhado aumenta o nível de satisfação do ECOS (Manikas, 2016).

O compartilhamento de elementos em um ECOS deve ser tratado considerando o conceito de transparência. Leite e Cappelli (2010) definem o escopo de transparência em três níveis: a transparência organizacional, a transparência direcionada e a transparência social, cada qual com um foco específico: a primeira foca os interessados, a segunda os consumidores e a terceira os cidadãos. Quanto maior a transparência, maior a qualidade de compartilhamento (Holzner e Holzner, 2006). Para tal, os principais interessados em um ECOS devem definir o escopo do seu interesse e o nível de transparência que deseja ver presente no ECOS (tipos de características). Utilizar os conceitos de transparência no enfoque de ECOS implica em aplicar as características da Figura 1 aos produtos e processos tratados no ecossistema. Essa aplicação vai requerer processos de negociação entre os atores e a implementação de operacionalizações adequadas à "satisfação a contento"[1] das referidas qualidades/características visíveis na Figura 1. Uma vez isso aceito como premissa, resta saber como responder a questões tais como: a) a quem se destina a transparência, ou seja, quais atores terão acesso ao compartilhamento de informações?, b) o que significa maior ou menor transparência? e c) que práticas deveriam ser implementadas para que um conjunto de atores seja transparente?

4. Agenda de Pesquisa para Transparência em Ecossistemas de Software

A partir da discussão promovida, esta seção propõe uma agenda de pesquisa inicial para explorar transparência em ECOS. Utilizou-se a categorização dos seis desafios de ECOS definida por Barbosa *et al.* (2013), relacionando-os com os conceitos da Figura 1. Muitos desses desafios são do desenvolvimento de software, mas afloram em ECOS devido à complexidade e à diversidade de interesses entre participantes neste universo.

Ecossistemas Abertos: é importante explorar como a característica *informativo* pode apoiar os ECOS abertos a manterem e expandirem o compartilhamento de informações acerca de recursos, artefatos e informações, bem como desenvolver métodos, técnicas e ferramentas para tratar isso em repositórios de software.

Governança: as características *entendimento* e *auditabilidade* podem ajudar na concepção e seleção de estratégias para assegurar a sustentabilidade da plataforma do ECOS, trazendo o desafio de como gerenciar e monitorar dependências de fornecedores, de tecnologias e de objetivos de negócio do ECOS, de forma mais transparente.

Análise: torna-se crítico operacionalizar as características *acessibilidade*, *usabilidade* e *entendimento*, uma vez que modelos, visualizações e grandes volumes de dados são usados para instrumentalizar o ECOS com o conhecimento necessário para tomada de decisões sobre parcerias ou admissão de membros.

Abertura: a partir da essência de um ECOS, o desafio de lidar com as características *acessibilidade* e *auditabilidade* surge como algo crítico, de modo que elementos que interferem ou afetam o sucesso da plataforma precisam ser melhor investigados, bem como permissões e níveis de acesso dos vários atores às informações compartilhadas.

Qualidade: a transparência é um requisito não funcional (qualitativo) de sistemas de software e, como tal, não é específica de ECOS, embora a forma como ela é tratada traz à tona a necessidade de explorar como a experiência de desenvolvedores e usuários afeta a qualidade geral do ECOS. Assim, as diversas características da Figura 1 contribuem para essa qualidade geral; *e.g.*, um estudo mapeando o modelo de qualidade MPS e a transparência de software aponta para algumas similaridades (Sousa et. al. 2015).

Arquitetura de Software: para apoiar o escopo e o nível de transparência acordados em um ECOS, é preciso uma arquitetura que atenda a essas necessidades e operacionalize as várias características. Portanto, diferentes operacionalizações para a Figura 1 devem ser levadas em consideração na arquitetura, *e.g.*: ser aberta, atender a padrões, tratar protocolos estabelecidos, ser fácil de aprender, ser usável, ser consistente e ser evoluída.

5. Considerações Finais

Este artigo apresentou uma discussão preliminar sobre transparência em ECOS e identificou desafios e oportunidades para a sua implementação. É preciso ter em mente

[1] Tradução do termo "satisfice" cunhado por Herbert Alexander Simon.

que a transparência da informação afeta o contexto de ECOS, uma vez que é necessário, por exemplo, assegurar transparência nos processos de negócio, no desenvolvimento dos sistemas ou mesmo nas informações disponibilizadas para agentes externos ao ECOS. Por meio deste estudo, pode-se perceber que, entre as características críticas da transparência para ECOS, *informativo* e *auditabilidade* referem-se respectivamente à gestão do conhecimento/aprendizagem no ECOS e ao gerenciamento/monitoramento da plataforma ao longo do tempo. Nesse sentido, considerando a ampla literatura de qualidade de software e a necessidade de tratar transparência em ECOS, adaptações de modelos e estratégias existentes para esse contexto pode ser um caminho. Além disso, a carência de pesquisas sobre transparência em ECOS abre oportunidades para trabalhos que analisem os desafios listados de maneira mais aprofundada.

Agradecimentos

Ao CNPq (Proc. No. PDJ 150539/20016-9) e a FAPERJ pelo apoio financeiro.

Referências

Barbosa, O. *et al.* (2013) “A Systematic Mapping Study on Software Ecosystems through a Three-dimensional Perspective”. In: Jansen, S. *et al.* (eds.) Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry, Edward Elgar Pub., 59-81.

Brasil (2011) “Lei nº 12.527, de 18 de novembro de 2011. Regula o acesso a informações previsto no inciso XXXIII do art. 5º, no inciso II do § 3º do art. 37 e no § 2º do art. 216 da Constituição Federal”. Diário Oficial da República Federativa do Brasil, Brasília, 18/11/11.

Cappelli, C. (2009) “Uma Abordagem para Transparência em Processos Organizacionais Utilizando Aspectos”. Tese de Doutorado. DI/PUC-Rio, Rio de Janeiro, Brasil.

Cataldo, M., Herbsleb, J.D. (2010) “Architecting in Software Ecosystems: Interface Translucence as an Enabler for Scalable Collaboration”. In: 4th ECSA, Copenhagen, 65-72.

Cruz, W., Maciel, C., Castilho, F., Girata, N. (2016) “Um Método Quantitativo para Avaliar a Adoção de Dados Abertos nos Tribunais de Contas do Brasil”. iSys Revista Brasileira de Sistemas de Informação 9(1):33-57.

Cukierman, H.L., Teixeira, C., Prikladnicki, R. (2007) “Um Olhar Sociotécnico sobre a Engenharia de Software”. Revista de Informática Teórica e Aplicada 14(2):207-227.

Fotrousi, F., Fricker, S.A., Fiedler, M., Le-Gall, F. (2014) “KPIs for Software Ecosystems: A Systematic Mapping Study”. In: 5th ICSOB, Paphos, 194-211.

Germano, C.E., Takaoka, H. (2012) “Uma Análise das Dimensões da Qualidade de Dados Abertos em Projetos de Dados Governamentais Abertos”. In: Anais do Congresso CONSAD de Gestão Pública, Brasília, 1-21.

Holzner, B., Holzner L. (2006) “Transparency in Global Change: The Vanguard of the Open Society”. University of Pittsburgh Press, 1st edition.

Leite, J.C.S.P., Cappelli, C. (2010) “Software Transparency”. Business & Information Systems Engineering 2(3):127-139.

Lord, K.M. (2006) “The Perils and Promise of Global Transparency”. SUNY Press.

Manikas, K. (2016) “Revisiting Software Ecosystems Research: A Longitudinal Literature Study”. JSS 117(2016):84-103.

Santos, R.P. *et al.* (2014) “Qualidade em Ecossistemas de Software: Desafios e Oportunidades de Pesquisa”. In: V CBSoft, VIII WDES, Maceió, v. 2, 41-44.

Sousa, H.P.S, Leal, A.L.C., Leite, J.C.S.P. (2015) “Alinhamento de Operacionalizações entre Transparência e MPS.BR”. iSys Revista Brasileira de Sistemas de Informação 8(4):109-141.

Criteria for Description of MDE Artifacts

Fábio P. Basso[1], Toacy C. Oliveira[1], Cláudia M. L. Werner[1], Valdemar V. Graciano Neto[2,3,4], Flavio Oquendo[4], Elisa Yumi Nakagawa[3]

[1]PESC - Federal University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil

[2]INF - Federal University of Goiás, Goiânia, GO, Brazil

[3]ICMC - University of São Paulo (USP), São Carlos, SP, Brazil

[4]IRISA-UMR CNRS - Université de Bretagne-Sud, Vannes, France

{fabiopbasso,toacy,werner}@cos.ufrj.br, valdemarneto@inf.ufg.br

flavio.oquendo@irisa.fr, elisa@icmc.usp.br

Abstract. *The large-scale collaboration in Model Driven Engineering (MDE) demands the use of one or more repositories such as GEMOC, ReMoDD and SEMAT, which share artifacts (e.g., models, meta-models, transformations, etc) for global reuse scenarios. While recent researches motivated the need of data for decision making from the perspective of Software Ecosystems (SECOs), the cutting edge technologies on this topic merely classify these artifacts. Likewise, it is important the definition of criteria to represent this qualified data with enough semantic potential to support reuse of MDE Artifacts globally, which is missed in the literature of the area. This paper proposes some criteria, thus a small contribution to help on the establishment of a future SECO for MDE.*

1. Introduction

Model Driven Engineering (MDE) is an approach that considers models as first class citizen in software development [Whittle et al. 2015]. Typical artifacts in MDE (termed as **MDE Artifacts**) often include models, metamodels, model transformations, model managers, and domain-specific models. MDE as Service (MDEaaS) is a novel approach for MDE that seeks to provide access to MDE through services [Basso et al. 2015]. Such artifacts are supposed to be stored and discovered from repositories or Knowledge Bases (KB) to be downloaded and integrated into software projects, as illustrates Figure 1.

In Phase 1, this approach can foster reuse of artifacts previously produced through repositories, reducing cost and time-to-market to integrate them in appropriate representations in Phase 3 [Mussbacher et al. 2014]. Repositories, to be effective, need a sort of semantic information (so-called *qualified data*) associated with such artifacts. These data shall describe applicability, guide the reuse of artifacts and allow the comparison between different features from artifacts in Phase 2, thus requiring well structured and qualified data. However, the literature does not properly offer solutions for this phase.

In a previous work, we claimed that the execution of software engineering services for reuse of MDE Artifacts in global scale can benefit from a perspective of Software Ecosystem (SECO) [Basso et al. 2015], whose represented

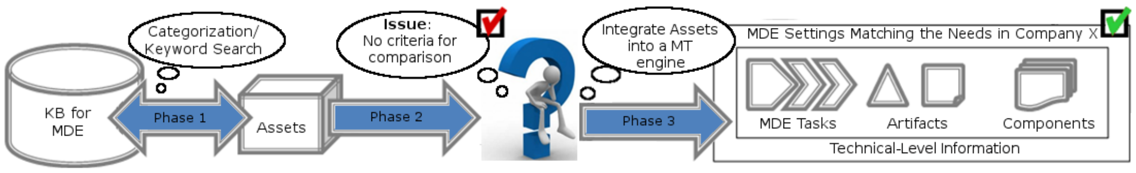

KB for MDE
Categorization/ Keyword Search
Phase 1
Assets
Issue: No criteria for comparison
Phase 2
Integrate Assets into a MT engine
Phase 3
MDE Settings Matching the Needs in Company X
MDE Tasks
Artifacts
Components
Technical-Level Information

Figure 1. A scenario for qualified descriptive data of MDE Artifacts.

assets are often analyzed under a three-dimension perspective (technical, business, and social) [Santos and Werner 2012]. Likewise, a first step towards the advent of a SECO for MDE is to meet reuse opportunities from repositories such as SEMAT [Jacobson et al. 2012], GEMOC [Combemale et al. 2014] and ReMoDD [France et al. 2007]. This requires the representation of assets with qualified data that describes and supports a three-dimension perspective of MDE Artifacts. However, assets are currently specified following an ad-hoc strategy, which hampers the execution of the second phase of Figure 1. Thus, our contribution is a set of criteria for representation of descriptive data for these assets.

This work is organized as follows: Section 2 provides background in repositories for MDE Artifacts; Section 3 proposes a criteria for representation of descriptive information associated with MDE Artifacts; Section 4 presents some insights for future works and Section 5 summarizes our conclusions.

2. Repositories for MDE Artifacts

MDE is often adopted to engineer domain-specific solutions. As such, many artifacts produced to comply with a subset of requirements in one project could be adopted again to develop other product which integrate the portfolio of a company. However, few is discussed about the quality of descriptive information to be associated with the shared artifacts, which is critical for decision making for one asset or other to be introduced in specific contexts [Whittle et al. 2015]. Figure 1 illustrates how this could be achieved considering the current availability of repositories in a possible implementation of a MDEaaS approach. Hence, artifacts could be searched, analyzed and integrated in new MDE projects.

Currently, existing work proposes the adoption of a unique underlying infrastructure (a unique KB) that fosters reuse of MDE Artifacts in Phase 1. Much of the effort in repositories is dedicated to tool support, focusing in classification [Lúcio et al. 2014]. REMODD is a repository for MDE that can be used to tackle the problem of reusing and sharing MDE Artifacts [France et al. 2007]. An initiative to accomplish a KB named SEMAT focuses on reusable methods [Jacobson et al. 2012]. SEMAT adopts Essence as a core representation language, which is useful to represent methods but limited to represent data from MDE technicalities. GEMOC [Combemale et al. 2014] is another proposal in this direction.

Instead this simplest approach, the execution of MDEaaS could consider the reuse of artifacts distributed in these several repositories. This needs a common representation language and the uniformity of information [Basso et al. 2015]. However, although some representation languages are available for representation of data

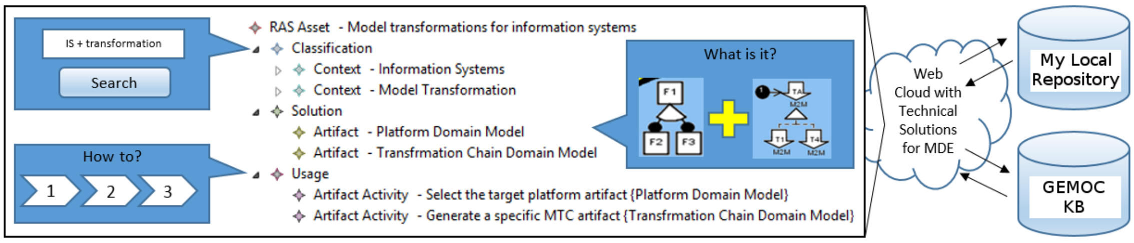

IS + transformation
Search
How to?
1
2
3
RAS Asset - Model transformations for information systems
Classification
Context - Information Systems
Context - Model Transformation
Solution
Artifact - Platform Domain Model
Artifact - Transfrmation Chain Domain Model
Usage
Artifact Activity - Select the target platform artifact {Platform Domain Model}
Artifact Activity - Generate a specific MTC artifact {Transfrmation Chain Domain Model}
What is it?
F1
F2
F3
TA
M2M
T1
T4
Web Cloud with Technical Solutions for MDE
My Local Repository
GEMOC KB

Figure 2. Information represented in a reusable asset conforms to RAS.

independently from repository providers [Basso et al. 2016], there is no criteria for representation of MDE assets. Besides, there is no requirements for data in support for SECO perspectives in MDEaaS. This limitation is tackled in this paper.

3. Criteria for Qualified Descriptive Data

Figure 2 exemplifies data associated with two MDE Artifacts in an asset. It is based on the Reusable Asset Specification (RAS) [Basso et al. 2016], a standard from the Object Management Group (OMG) for externalize data associated with artifacts from reuse repositories, i.e., making an asset independent from a repository vendor (e.g., My Repository or GEMOC KB). An asset conforms to RAS allow the use of some structures for descriptive data: 1) *Classification*, allowing the automatic cataloging in repositories; 2) *Solution*, allowing the detailing of data from artifacts such as type, variability points and semantics; and 3) *Usage*, allowing the representation of instructions for reuse after acquisition/download.

Motivation: Instead of the imposition of use of a unique KB, we found relevant the use of them all through a common representation. Likewise, we need to reach criteria for representation of information for assets independently from a repository [Basso et al. 2015]. Although RAS plays a similar role as layers in a network protocol that are processed in levels of abstraction, it is also important use criteria for representation of data associated with MDE Artifacts. Thus, we present criteria for data used in Phase 2 considering technical, business, and social needs.

Criteria formulation: It was conducted an analytical study of representativeness of two asset specification languages [Basso et al. 2016]. We also looked in the literature based in a structured keyword search for recommendations to descriptive data, but we missed focused papers that could be used. Then, we conducted a multivocal literature review [Ogawa and Malen 1991], which is mainly characterized by the inclusion of material not available in search engines. The following source of information is investigated in a multivocal research protocol: a) searched in books about design patterns and empirical software engineering; b) searched in papers on management of model-based operations; and c) looked for calls for contributions (tool demo) from conferences related with the MDE (MODELS, ECMFA, SPLC, GPCE, SLE, etc.) and in more general conferences (ASE, ICSE, OOPSLA).

Criteria for the descriptive-level of MDE Artifacts:

- **Information about the context.** It is important to represent information that answer some questions recommended in book for design patterns [Gamma et al. 1995]: a) What is the solution? b) How to use it? c) Who can use it? d) Known uses? e) Which are the known incompatibilities? f) Which is the software license and its dependencies? g) Examples, etc.

- **Information about the application.** Questions recommended by empirical engineering [Whittle et al. 2015, Mussbacher et al. 2014] are important to establish comparisons between competitor MDE Artifacts such as: a) Which are the minimum requirements to use an MDE Artifact including teams configuration, members skills (social and technical) which the artifact have been used successfully, the required knowledge from end-users about domain specific languages, with which software process and so on? b) Which are the pre and post-conditions to use it? c) Which are the evidences that support the solution and in which boundary conditions? d) Which is the learning curve and the Return on Investment (ROI)? and e) Feedback from users.
- **Quality for the descriptive information.** The following questions should be considered when representing descriptive data from MDE Artifacts: a) Is the information for catalog based in standard taxonomy for searching technical solutions? b) Is this standard taxonomy represented in a KB as data dictionary, thus describing what means the contexts used in catalog information? c) Is clear the instructive information such as how to integrate and use artifacts in contexts?

Criteria for descriptive and technical-level data:

- **Structure for the descriptive information.** The following information should be structured to enable analysis of MDE Artifacts: a) Abstractions for types of artifacts produced and consumed in software processes phases by a tool in the context of MDE; b) Abstractions that makes clear when an MDE Artifact needs adaptation before introduction in a target context; c) Textual information for decision making (business, technical and social perspectives), which could be based on structures for design patterns, so one can decide the option that best meet a specific need in a software development context.
- **Link to technical information.** Descriptive information itself is useless, which means that technical information should also be considered such as semantics to: a) The target platform associated with transformations for model-to-text/model-to-code/code-to-model; b) The type of the model transformation [Lúcio et al. 2014]; c) The meta-models and the meta-model framework [Combemale et al. 2014], as well as the design languages associated with each model transformation, such as UML Profiles, or Graphical DSL, or Textual DSL; e) Model transformation components, bindings and parameter matching; and f) Variability points [Basso et al. 2016].

4. Ongoing and Future Works

Figure 2 shows the structures for representation data for MDE Artifacts, which is interesting as a common language, but limited when applying the proposed criteria. In order to enable the implementation of the scenario illustrated in Figure 1, future works shall further investigate the following aspects: a) New abstractions for a *Classification* structure, which allows the storage and search of MDE Artifacts in a KB with data about context and application (Phase 1); b) Structures for design patterns, used for representation of data to support decision making (Phase 2); and c) Abstractions for technicalities in settings, represented as instances of *Artifact* for MDE specificity (e.g., chains of components) connected with a structure called *Usage* (Phase 3).

5. Final Remarks

This paper address a set of criteria with requirements for representation of descriptive data of MDE Artifacts. These artifacts are currently available in repositories for MDE allowing (i) support their use through services, (ii) facilitate their discovery and understanding of such artifacts, and (iii) foster their systematic reuse. To be effective, it is necessary to represent qualified descriptive data, which add semantic to such artifacts, thus promoting their adoption in new MDE software development projects. So far, it is not clear what must be represented in this level of abstraction considering a SECO perspective. Thus, through presented criteria and work done, this perspective can now be introduced in MDE as Service based in a common format for representation of assets and in recommendations for descriptive data.

References

Basso, F. P., Oliveira, T. C., and Werner, C. M. L. (2016). Analysis of asset specification languages for representation of descriptive data from mde artifacts (to appear). In *Conference on ENTERprise Information Systems, October 5-7*.

Basso, F. P., Werner, C. M. L., and Oliveira, T. C. (2015). A summary of challenges for "mde as service". WDES'15, pages 85–88, Belo Horizonte-MG, Brazil.

Combemale, B., Deantoni, J., Baudry, B., France, R., Jézéquel, J.-M., and Gray, J. (2014). Globalizing modeling languages. *IEEE Computer*, 47(6):68–71.

France, R., Bieman, J., and Cheng, B. (2007). Repository for model driven development (remodd). 4364 LNCS, pages 311–317.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns, Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Jacobson, I., Ng, P., McMahon, P. E., Spence, I., and Lidman, S. (2012). The essence of software engineering: The semat kernel. a thinking framework in the form of an actionable kernel. *ACMQueue. Development 9. Networks*, 10(10):1–12.

Lúcio, L., Amrani, M., Dingel, J., Lambers, L., Salay, R., Selim, G. M., Syriani, E., and Wimmer, M. (2014). Model transformation intents and their properties. *Software & Systems Modeling*, pages 1–38.

Mussbacher, G., Amyot, D., Breu, R., Bruel, J.-M., Cheng, B. H., Collet, P., Combemale, B., France, R. B., Heldal, R., Hill, J., Kienzle, J., Schöttle, M., Steimann, F., Stikkolorum, D., and Whittle, J. (2014). The relevance of model-driven engineering thirty years from now. In *MODELS*, pages 183–200.

Ogawa, R. T. and Malen, B. (1991). Towards rigor in reviews of multivocal literatures: Applying the exploratory case study method. *Review of Educational Research*, 61(3):265–286.

Santos, R. P. d. and Werner, C. (2012). Reuseecos: An approach to support global software development through software ecosystems. ICGSEW, pages 60–65, Washington, USA. IEEE.

Whittle, J., Hutchinson, J., Rouncefield, M., Håkan, B., and Rogardt, H. (2015). A taxonomy of tool-related issues affecting the adoption of model-driven engineering. *Software & Systems Modeling*, pages 1–19.

Descrição de um Ecossistema de Software para um Ambiente Virtual de Aprendizagem: ECOS SOLAR

Emanuel F. Coutinho[1], **Ítalo de Oliveira**[1], **Carla I. M. Bezerra**[2]

[1]Instituto Universidade Virtual (IUVI) – Fortaleza – CE
[2]Campus Quixadá – Quixadá – CE
Universidade Federal do Ceará (UFC) – CE – Brasil

{emanuel,italo}@virtual.ufc.br,carlailane@ufc.br

Abstract. *A software ecosystem (ECOS) refers to a set of software with a certain degree of symbiotic relationship. Virtual Learning Environments (VLE) integrate technology of information and communication, aiming to create environments based on the Internet to enable the process of building knowledge and autonomy from their interactors. The aim of this work is to allow an overview of the ECOS SOLAR and some research challenges.*

Resumo. *Um Ecossistema de Software (ECOS) refere-se a um conjunto de produtos de software com determinado grau de relacionamento simbiótico. Ambientes Virtuais de Aprendizagem (AVA) integram TICs, visando a criação de ambientes na Internet que possibilitem o processo de construção de conhecimento e autonomia por parte de seus interagentes. O objetivo deste trabalho é apresentar uma visão geral do ECOS SOLAR e alguns desafios de pesquisa.*

1. Introdução

Um ECOS consiste em um conjunto de atores agindo como uma unidade que interage com um mercado distribuído entre software e serviços, juntamente com as relações entre estas entidades [Jansen et al. 2009]. Tais relações são frequentemente apoiadas por uma plataforma tecnológica ou por um mercado comum e realizadas pela troca de informação, recursos e artefatos. O SOLAR (Sistema *Online* de Aprendizagem) [Solar 2016] é um Ambiente Virtual de Aprendizagem (AVA), desenvolvido pelo Instituto Universidade Virtual (IUVI), da Universidade Federal do Ceará (UFC). É uma aplicação *web* de três camadas, cujo modelo de participação é orientado ao professor e ao aluno, possibilitando a publicação de cursos e interação com os mesmos. O objetivo principal deste trabalho é apresentar o ecossistema de software do AVA SOLAR. Como objetivo secundário, pretende-se apresentar alguns desafios de pesquisa relacionados ao ECOS SOLAR.

2. ECOS SOLAR

O ECOS SOLAR é composto por um conjunto de elementos que se comunicam de diversas formas. Esses elementos envolvem diferentes instituições com relacionamentos, produzindo ou recebendo informações, suportados por diferentes tecnologias.

O ECOS SOLAR é composto por: produtos do AVA SOLAR (versões *web*, *mobile* e MOOC - *Massive Open Online Course*), fornecedores (diferentes sistemas de controle acadêmico da universidade), concorrentes (diferentes AVAs, como o MOODLE), clientes e parceiros (UFC, Universidade Aberta do Brasil - UAB, IUVI, cursos de graduação, órgãos estaduais e sistemas gerenciais), setor de desenvolvimento (administração de dados, gerência de projetos, *designers*, desenvolvimento e produção didática), pesquisa e inovação (*Text-To-Speech* e laboratórios virtuais), sociedade (interiorização e inclusão digital) e oportunidades de pesquisa (acesso à base de dados, aplicação de pesquisas, comitê de pesquisa e ética).

3. Oportunidades de Pesquisa

Diversos desafios estão presentes no ECOS SOLAR, além de sua própria modelagem. É importante buscar o gerenciamento de qualidade entre todos os componentes que estão envolvidos no ecossistema. Para o ECOS SOLAR, por mais que os processos sejam desenvolvidos de forma independente, é necessário que o gerenciamento garanta a qualidade nos produtos de software que estão sendo desenvolvidos, trazendo para os desenvolvedores a aplicação de boas práticas durante o desenvolvimento, garantindo que o produto final seja consistente e consiga se integrar com os outros elementos do ECOS. A própria evolução dos produtos derivados do SOLAR deve ser tratada com cuidado, exigindo grande esforço de concepção, manutenção e evolução.

Outra forma de explorar melhor os elementos do ECOS SOLAR é entender o ponto de vista dos diversos indivíduos que o utilizam. Com isso, pretende-se buscar a melhor forma de agregar as perspectivas e conhecimentos de cada usuário, com o intuito de aprimorar o ECOS para suprir as necessidades dos usuários e clientes, possibilitando requisitos funcionais e não funcionais adequados e viáveis para seu desenvolvimento.

4. Conclusão

O AVA SOLAR atinge uma grande variedade de perfis (clientes, desenvolvedores, alunos, instituições, fornecedores), estando disponível atualmente em versões *web* e para dispositivos móveis. Este trabalho apresentou de maneira preliminar o ECOS SOLAR, descrevendo de maneira geral seus diversos componentes e instituições.

De maneira geral, os principais desafios de pesquisa do ECOS SOLAR são: (i) prover uma arquitetura de referência que suporte a reutilização de componentes de software; (ii) manutenção e evolução dos diversos produtos derivados do SOLAR e seus respectivos componentes de software, de maneira a serem orientados a reuso; e (iii) identificar e documentar os interesses dos parceiros e incoporá-los à plataforma tecnológica, com foco em inovação. Outra grande oportunidade de pesquisa é como evoluir as versões do SOLAR, adicionando novos componentes originados de pesquisa e inovação e como disponibilizá-los para os usuários de versões diferentes, alinhados a boas práticas de Engenharia de Software.

Referências

Jansen, S., Brinkkemper, S., e Finkelstein, A. (2009). Business network management as a survival strategy: A tale of two software ecosystems. In *Proceedings of the First International Workshop on Software Ecosystems, 11th International Conference on Software Reuse*, pages 34–48.

Solar (2016). Solar. http://www.solar.virtual.ufc.br/. Online; acessado em abril-2016.

Fatores de Influência na Experiência do Desenvolvedor em Ecossistemas de Software Móvel

Joicilene Santos[1], Priscila Silva Fernandes[1], Edson S. Gonçalves[1], Awdren Fontão[2], Bruno A. Bonifácio[1]

[1]Instituto de Ciências Exatas e Tecnologia (ICET) - UFAM
[2]Instituto de Computação (IComp) - UFAM

{joicilene, pry.bila, edsonserrao24, brunnoboni}@gmail.com, awdren@icomp.ufam.edu.br

Abstract. *The applications' production process also has a strong dependence on the human factor. Studies fall under the existence of factors that can affect the productivity of MSECOs' developers. This paper presents a preliminary study where factors can be identified that may impact on the developer experience in a MSECO.*

Resumo. *O processo de produção de aplicações ainda possui forte dependência do fator humano. Estudos relevam a existência de fatores que podem afetar a produtividade de desenvolvedores.O presente artigo apresenta uma proposta para identificação de fatores que podem impactar na experiência do desenvolvedor em MSECOs.*

1. Introdução

A popularização dos dispositivos móveis tem alterado a forma como as aplicações móveis (apps) têm sido produzidas [Coutinho *et al.,* 2015]. As organizações (e.g. Apple, Google e Microsoft) passaram a enfrentar diversos problemas no desenvolvimento, relacionados com o desempenho do desenvolvedor. Por essa razão, as organizações criaram novas estruturas e modelos de negócio para suportar as contribuições para os seus ecossistemas, como: criação de portais de apoio [Lim e Bentley, 2012].

Por essa razão, torna-se importante identificar características e fatores que podem interferir na Experiência do Desenvolvedor (DX), pois tais fatores podem auxiliar na melhoria do desempenho do desenvolvedor [Fontão *et al.,* 2015]. Nesse contexto, o presente trabalho tem como foco a caracterização inicial de fatores que influenciam na DX.

2. Caracterizando Fatores que Influenciam na DX em MSECO

A Experiência do Desenvolvedor (DX, do inglês *Developer Experience*) está ligada a qualquer tipo de envolvimento do desenvolvedor, recorrente a qualquer um que esteja comprometido a atividade de desenvolvimento de software [Fagerholm; Munch *et al.*, 2012]. Enquanto a meta do desenvolvedor é criar software, a DX considera aspectos relacionados ao sentimento, o valor e o efeito sobre artefato produzido. Vários estudos tem focado em DX na literatura [Trendowicz e Munch, 2009], [Paiva *et al.* 2010] e [Ribeiro *et al.,* 2015].

Um estudo experimental inicial foi realizado sob o ponto de vista dos desenvolvedores, comparando dois MSECOs: o *Android* e o *Windows Phone.* Para avaliação dos fatores identificados conforme a literatura foi aplicado questionário a vinte desenvolvedores, 10 desenvolvedores *Android* e 10 *Windows Phone.* Como resultado foi identificado que para os vinte e dois fatores avaliados entre os dois MSECOs selecionados, não há diferença na percepção dos desenvolvedores de acordo a avalição dos mesmos. Os fatores e a porcentagem de influência são apresentados nas Figuras 1 e 2. Conforme os resultados da avaliação da pequena amostra utilizada neste estudo há indícios que tais fatores podem influenciar no desempenho dos desenvolvedores seja de forma positiva ou negativa. Assim, espera-se com estes resultados incentivar a realização de novas pesquisas e avaliações de fatores que podem influenciar na DX.

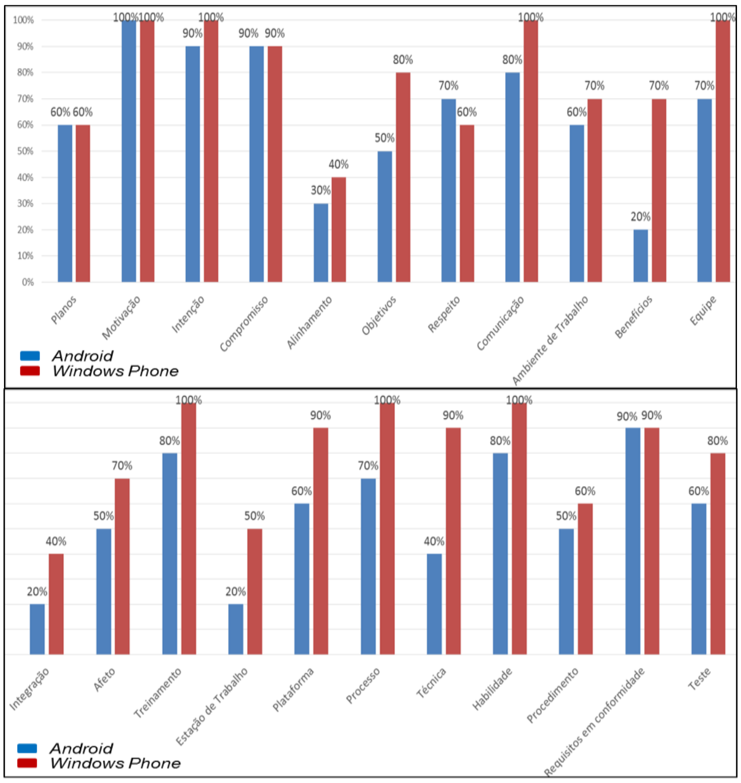


Figura 1. Fatores e Porcentagem de Influência.

Referências

Coutinho, W.; Couto, E.; Biase, C.; Fernandes, P.; Bonifacio, B. 2015. Improving An Educational Mobile Application Through Usability Evaluation. In 9th International Technology, Education and Development Conference. Proceedings, Spain. p.5812-5820.

Fagerholm, F.; Münch, J. (2012). Developer experience: Concept and definition. International Conference on Software and System Process.

Fontão, A., Santos, R., Dias-Neto, A. 2015. Research Opportunities for Mobile Software Ecosystems. Proceedings of the 9th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (WDES), Belo Horizonte, 97–98.

Lim, S. L.; Bentley, P. J. 2012. How to be a successful app developer: lessons from the simulation of an app ecosystem. ACM New York, p. 129–136.

Paiva, E.; Barbosa, D.; Lima, R.; Albuquerque, A. 2010 Fatores que influenciam a produtividade de desenvolvedores de software na visão dos gerentes de projeto.In: IV Conference on Research and Practical Issues of Enterprise Information Systems p. 2-12.

Ribeiro, D., França, C. e Pinto, P. 2010 Motivação dos Engenheiros de Software no Contexto Open Source: Um Estudo de Caso com a Comunidade Android Brasil Projetos.

Trendowicz, A.; Münch, J. 2009. Factors Influencing Software Development Productivity-State-of-the-Art and Industrial Experiences.Advances in computers, v. 77, p. 185-241.

Um Roteiro para a Classificação dos Papéis dos Atores em Ecossistema de Software no Contexto Público

Rebeca Teodoro da Silva[1, 2], Luiz Gustavo Ferreira Aguiar[1,2], Elias Canhadas Genvigir[1]

[1]Universidade Tecnológica Federal do Paraná (UTFPR) – Cornélio Procópio – PR

[2]Tribunal de Justiça do Paraná (TJ PR)

{rebeca.teodoro, gustavo.bytes}@gmail.com, elias@utfpr.edu.br

Abstract. *This paper shows a roadmap for the roles actors classification in ECOS based on their relationships to assist in understanding existing relations in the ECOS.*

Resumo. *Neste artigo é apresentado um roteiro para a classificação dos papéis dos atores em ECOS baseado nos seus relacionamentos, com o intuito de auxiliar a compreensão das relações existentes no ECOS.*

1. Introdução

Um ECOS consiste basicamente de elementos como um centralizador, uma plataforma que pode ser uma tecnologia ou o mercado e os agentes do nicho relacionado (Pereira et al., 2013). A intensidade da interação entre atores e artefatos no ECOS pode levar à fusão das redes sociais e das redes técnicas, resultando em uma rede socio-técnica, que abrange elementos e associações de ambas as redes (Lima, Barbosa, Pereira, & Werner, 2014).

Este artigo apresenta um roteiro para a classificação dos papéis dos atores existentes em ECOS do contexto público baseado nos relacionamentos e interações. O objetivo do roteiro é possibilitar uma visão geral dos papéis dos atores que estão relacionados com a saúde do ECOS, potencializando a compreensão das relações envolvidas no ECOS.

2. Metodologia

Foi realizado um estudo de caso que, segundo Yin (Yin, 2014), é um estudo empírico que investiga um fenômeno atual dentro do seu contexto de realidade, quando as fronteiras entre o fenômeno e o contexto não são claramente definidas e no qual são utilizadas várias fontes de evidência. Assim, foi selecionado o ECOS Projudi do Tribunal de Justiça do Paraná. Para a coleta dos dados foi utilizado entrevistas, realizadas no ano de 2016, com funcionários da área de TI, como analistas de sistemas que possuem contato com o software keystone selecionado para o estudo. Em relação às interações o Projudi possui várias interações, visando fins específicos, entre o sistema e os perfis de atores envolvidos. Além disso, alguns sistemas são interligados ao sistema Projudi no Tribunal de Justiça do Paraná – TJPR (Silva, Ferreira, & Genvigir, 2015).

Como pergunta objeto tem-se "como realizar a classificação dos papéis dos atores de um ECOS no contexto público? ". Para responder esta pergunta foi elaborado um roteiro para auxiliar classificar os papéis dos atores envolvidos no ECOS.

2. Resultados

Na Figura 1 é apresentado o roteiro que pode ser utilizado para auxiliar na classificação dos papéis dos atores que estão envolvidos no ECOS do contexto público baseados nos seus relacionamentos.

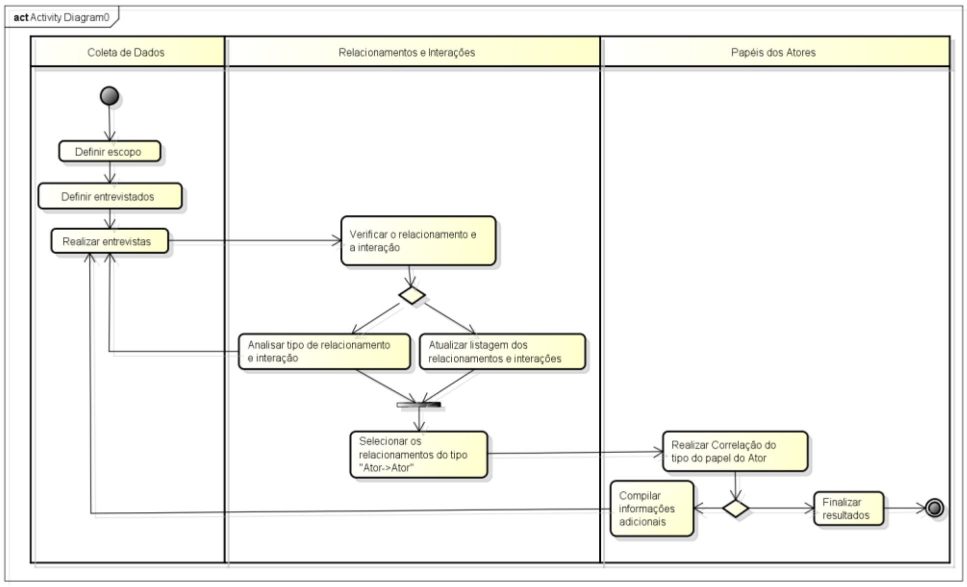

act Activity Diagram0
Coleta de Dados
Relacionamentos e Interações
Papéis dos Atores
Definir escopo
Definir entrevistados
Realizar entrevistas
Verificar o relacionamento e a interação
Analisar tipo de relacionamento e interação
Atualizar listagem dos relacionamentos e interações
Selecionar os relacionamentos do tipo "Ator->Ator"
Realizar Correlação do tipo do papel do Ator
Compilar informações adicionais
Finalizar resultados

Figura 1 - Roteiro para a classificação dos atores de ECOS no contexto público

Ao analisar os relacionamentos existentes no ECOs Projudi, observa-se que este pode ser classificado em 2 principais papéis dos atores envolvidos: *Niche Player – Customer* e *Niche Player – Developer – Influencer*. Dado o seu contexto público os relacionamentos e interações presentes com outros atores do ECOS são influenciados pelo nicho de negócio no qual está inserido, no entanto são determinados por leis, hierarquias e políticas públicas.

O roteiro apresentado pode ser utilizado como referência para classificar os papéis dos atores envolvidos no ECOS, baseado nos seus relacionamentos. Assim, esta pesquisa ajuda na compreensão teórica da influência dos papéis dos atores em um ECOS no contexto público.

3. Referências

Lima, T., Barbosa, G., Pereira, R., & Werner, C. (2014). Uma Abordagem Socio-técnica para Apoiar Ecossistemas de Software, *7*(3), 19–37.

Pereira, R., Maria, C., Werner, L., Alves, C. F., Jorge, M., Pinto, S., … Egler, C. (2013). Ecossistemas de Software : Um Novo Espaço para a Construção de Redes e Territórios envolvendo Governo, Sociedade e a Web. *Políticas Públicas: Interações E Urbanidades*, *1ed*, 337–366.

Silva, R. T., Ferreira, L. G., & Genvigir, E. C. (2015). Ecossistema de Software no Contexto do Poder Judiciário - Apontamentos Sobre o Estado do Paraná. *9thWORKSHOP ON DISTRIBUTED SOFTWARE DEVELOPMENT, SOFTWARE ECOSYSTEMS AND SYSTEMS-OF-SYSTEMS*, *01*, 49–56.

Yin, R. k. (2014). *Estudo de Caso - Planejamento e método*. *Igarss 2014*. doi:10.1007/s13398-014-0173-7.2

Identificando Características de Aplicações para Simulação de Ecossistemas de Software Móveis

Allan Bezera[1,2], Awdren Fontão[1]

[1]Instituto de Computação – Universidade Federal do Amazonas (ICOMP-UFAM)
69.007-000 – Manaus – AM – Brasil

[2]Samsung Ocean Manaus – Universidade do Estado do Amazonas (OCEAN-UEA)
69.050-020 – Manaus – AM – Brasil

{allan.bezerra, awdren}@icomp.ufam.edu.br, allan.jsb@oceanbrasil.com

Abstract. *In a Mobile Software Ecosystem (MSECO), manufacturers provide a platform for developers to create mobile applications. A variety of factors may influence the success and performance of an application. In this context, this paper proposes a set of features identified through literature, reports and technical documentation. This characterization enables the creation of a MSECOs simulation.*

Resumo. *Em um Ecossistema de Software Móvel (MSECO), fabricantes disponibilizam uma plataforma para desenvolvedores criarem aplicações móveis. Uma diversidade de fatores pode influenciar no sucesso e desempenho de uma aplicação. Nesse contexto, este trabalho propõe um conjunto de características identificadas através da literatura técnica, relatórios e documentação técnica. Essa caracterização possibilita a criação de um ambiente de simulação interativo para MSECO.*

1. Introdução

A presença constante dos celulares no cotidiano das pessoas trouxe um aumento significativo na demanda por Aplicações Móveis (*Apps*), criando uma rede de criação e fornecimento de software chamado de Ecossistema de Software (MSECO).

O conhecimento das características que impactam diretamente na popularidade de uma aplicação é primordial para que se projete *Apps* melhores, sob a perspectiva do usuário. Assim, o entendimento da dinâmica envolvendo MSECO pode ser um diferencial entre o sucesso ou o fracasso de uma *App*.

Com isso, o uso de simulação mostra-se como uma potencial ferramenta para apoio à tomada de decisão. Em (BEZERRA et al., 2016), foi apresentado um mapeamento sistemático que caracterizou tais simuladores, dentre eles o modelo *AppEco* (LIM e BENTLEY, 2012). No entanto, este modelo não lista e enumera quais características seriam essas, ele as tratas simplesmente como números, o que dificultaria a sua aplicação em situações reais de desenvolvimento de *Apps*.

2. Característica de Aplicações para Simulação de MSECO

Diante deste problema, esse trabalho busca oferecer uma lista de características que descrevem uma *App*, sua classificação nas categorias Técnica, Social e de Negócio, conforme taxonomia apresentada em (SANTOS et al., 2011), mapeando assim os fatores determinantes para o sucesso de uma aplicação móvel, e permitindo assim a extensão do modelo teórico *AppEco*.

O trabalho de (LIM et al., 2015) traz uma pesquisa global extensiva com usuários de Aplicações de diversas partes do mundo, mapeando os aspectos gerais que influencia o usuário no momento de realizar o *download* de uma *App*. Para compor os aspectos apresentados por (LIM et al., 2015), esse trabalho realizou composição com outras fontes para chegar a proposição de uma lista mais completa de características.

As fontes selecionadas foram trabalhos disponíveis na literatura com notório reconhecimento técnico, documentação técnica das plataformas de MSECO e Relatório de desempenho das *Apps* disponibilizados pelos mantenedores das Lojas de Aplicativos, destacando estratégias de negócio de desenvolvedores de sucesso. A lista completa das características pode ser acessada através do endereço: https://goo.gl/Vx7o89

Mapear essas características viabilizará o oferecimento de um ambiente computacional baseado em simulação que permita, por meio da extensão do simulador *AppEco*, oferecer uma ferramenta que apoie a simulação do desempenho e evolução de uma *App*.

3. Conclusões e Trabalhos Futuros

Neste trabalho, foi apresentado um estudo para identificação de características que descrevem a *App*, mapeando assim os fatores determinantes para o sucesso uma aplicação móvel. Isso permite a extensão do modelo teórico *AppEco* e a aplicação prática em MSECOs. Como trabalho futuro, os resultados encontrados nesse estudo serão validados com um conjunto real de aplicações móveis e por meio de consulta a especialistas em MSECO.

Agradecimentos

Os autores agradecem à UFAM, UEA e SAMSUNG pelo apoio para a realização deste trabalho. Parte dos resultados apresentados neste artigo foi obtida por meio de atividades de P&D do projeto SAMSUNG OCEAN, patrocinado pela Samsung Eletrônica da Amazônia Ltda., sob os termos da lei federal No. 8.248/91.

7. Referências

BEZERRA, A.; FONTÃO, A.; DIAS-NETO, A. **Simulação de Ecossistemas de Software Móvel: Estado da Arte , Desafios e Oportunidades**XIX Congresso Ibero-Americano em Engenharia de Software - CIbSE 2016. **Anais**...2016

LIM, S. L. et al. Investigating Country Differences in Mobile App User Behavior and Challenges for Software Engineering. **Software Engineering, IEEE Transactions on**, v. 41, n. 1, p. 40–64, 2015.

LIM, S. L.; BENTLEY, P. J. **How to be a Successful App Developer: Lessons from the Simulation of an App Ecosystem**GECCO '12 Proceedings of the 14th annual conference on Genetic and evolutionary computation. **Anais**...2012

SANTOS, R.; MARIA, C.; WERNER, L. **A Proposal for Software Ecosystems Engineering**Proceedings of the Workshop on Software Ecosystems 2011. **Anais**...2011