1
Introduction

Model composition plays a central role in many software engineering activities, e.g., evolving design models to add new features (Thaker et al., 2007; Jayaraman et al., 2007) and reconciling models developed in parallel by different development teams (Wagner et al., 2003; Perry et al., 1998; Berzins, 1994). In fact, developers use model composition throughout the software development process, from the initial stage by integrating abstract design models (e.g., conceptual models) to the final stage by composing more detailed ones (e.g., UML class and sequence diagrams). In collaborative software development, for example, separate development teams may concurrently work on specific parts of an overall design model that are more relevant to them. However, it is necessary at some point to bring these models together in order to create a “big picture view” of the overall design model. For this reason, to date, there has been a significant body of research about model composition in the areas of model management (IBM, 2012), integration of software product lines (Jayaraman et al., 2007), and software merge (Mens, 2002).

The term *model composition* can be briefly defined as a set of tasks that should be performed to combine two (or more) input models, M_A and M_B, in order to produce an output intended model, M_{AB} (Brunet et al., 2006; Mens, 2002; Clarker, 2001). However, an output composed model, M_{CM}, is usually produced instead of M_{AB}. While the M_{CM} would be the model produced by a model composition technique, the M_{AB} is, in fact, the model intended by developers. The M_{CM} often needs to be reviewed and changed to become compliant with M_{AB}. These models seldom match ($M_{CM} \neq M_{AB}$) as some properties of the M_A and M_B conflict with each other. If not properly handled, these conflicts may cause syntax and semantic inconsistencies in M_{CM}. Therefore, in order to transform M_{CM} into M_{AB}, developers must also invest effort to identify and resolve these inconsistencies.

In practice, developers use model composition if they understand the effort to obtain M_{AB}. However, developers are unable to grasp the composition effort and realize any cost-effectiveness analysis. Hence, they are left without any practical knowledge about the effort to be invested in order to compose the design models apart from evangelists' anecdotal feedback, which often diverge from each other. If model composition is an error-prone and effort-consuming activity, then the potential benefits, e.g., gains in productivity, can be compromised. This inability of evaluating composition effort is due to three problems. First, the current measurement approaches are inadequate to assess the concepts found in model composition, such as specific effort dimensions, conflicts, and inconsistencies. Second, researchers and developers do not know the factors that can influence the composition effort in practice. Examples of key factors would be: (i) the design decomposition (e.g., object-oriented design or aspect-oriented design) represented by a certain modeling language, and (ii) the selected composition technique (e.g., IBM Rational Software Architecture) that is responsible for supporting the composition of design models. Third, practical knowledge about how the influential factors may affect the developers' effort is severely lacking. To date, there exists a clear need for addressing these problems as software modeling is increasing collaborative (France & Rumpe, 2007). If the effort on model composition is high, then the potential benefits (e.g., effectiveness in producing M_{AB}) of using model composition can be hindered in real projects.

It is important to address these problems due to several other reasons. First, before adopting, for example, a model composition technique in practice, developers need appropriate evaluation frameworks to reveal the actual effort to obtain M_{AB} in practical settings. This decision should be supported by practical knowledge rather than evangelists' estimation. Second, by knowing the influential factors on model composition effort, they can make decisions more effectively. For example, at the early stages of software projects, developers need to choose which design decomposition will be used (e.g., object-orientation or aspect-orientation), which design characteristics will be applied to the design models (e.g., stability), and which composition technique will be adopted (e.g., IBM RSA or Epsilon). In addition, developers can reduce side effects of such decisions if they can rely on such knowledge up front. For example, developers can use a particular type of composition technique in software evolution scenarios where they are known to be more cost-effective than others can. Third, by empowering researchers with lessons learned from empirical studies, they can precisely improve existing modeling languages and composition techniques, thereby reducing the error likelihood and effort of composing design models.

With these issues in mind, it is particularly important, albeit challenging, to measure effort and understand the factors that can jeopardize the composition of design models. The definition of software metrics and the execution of empirical studies have been pointed out as a powerful way to gather empirical evidence in software engineering fields (Fenton & Pfleeger, 1997) as well as to derive lessons learned (Kitchenham et al., 2008; Wohlin et al., 2000). The remainder of this Chapter is organized as follows. Section 1.1 presents the problem statement. Section 1.2 describes the limitations of the related work. Section 1.3 describes the study methodology. Section 1.4 elaborates the key contributions of this thesis. Finally, Section 1.5 describes how the next chapters are organized.

1.1. Problem Statement

The problem of empirical evaluation of model composition effort is rooted in the inadequate support for measuring this effort and the lack of practical knowledge to design empirical studies in this context. In fact, current studies on model composition neither explicitly take into account effort as a measurement unit nor even provide indicators about how developers invest effort in practice. The current measurement methods for software design aim at simply quantifying specific properties of object-oriented (OO) decompositions (such as, degree of inheritance) and general properties of design models (e.g., coupling and cohesion), thereby failing to provide effective indicators for model composition effort. For example, from a sequence of output composed models, developers should be able to identify those models that are likely to have a high concentration of inconsistencies, which require a higher effort to produce the intended model. Indicators can help developers to identify those critical models.

Unfortunately, researchers are unable to properly evaluate model composition efforts nowadays. Hence, developers often make misinformed decisions without empirical knowledge about factors affecting model composition effort. For instance, the effort of applying a particular composition technique to compose UML models might be higher depending on the type of software change being realized. In addition, it might be that the composition effort of more modularized models might be substantially reduced. If so, this means that developers should invest more effort on improving the modularity of input design models before they are composed. If empirical knowledge of these factors is not available, designers are likely to invest much higher effort than what is needed when carrying out model composition. They are also likely to spend undesirable effort to detect and resolve inconsistencies because of misinformed decisions.

In addition, before adopting model composition in practice, it is necessary to have actual evidence of the effort that developers should invest to compose design models. The lack of appropriate measurement approaches jeopardizes the execution of empirical studies. In other words, without experimental investigations, model composition cannot be widely accepted in practice. This means that researchers are unable to properly test hypotheses, analyze correlations between variables, and perform comparative analysis of two or more empirical studies. Then, it is not possible to create a credible body of knowledge on composition effort supported by empirical evidence.

These shortcomings become more apparent in an age that model composition is starting to play a central role in many software engineering activities. In fact, model composition techniques are essential to support the evolution of design models in order to add new features (Thaker et al., 2007; Jayaraman et al., 2007) and reconcile models developed in parallel by different development teams (IBM, 2011; Wagner et al., 2003; Perry et al., 1998; Berzins, 1994). Unfortunately, model composition may become an effort-consuming task as the lack of knowledge about the influential factors (such as type of composition technique, design modeling language, and design characteristic) can bring harmful effects to the composition effort. The absence of a cost-effectiveness analysis, supported by effort indicators and experimental investigations, makes challenging the activity of composing design models. Therefore, researchers and developers need guidance for assessing model composition effort quantitatively and qualitatively.

1.2. Limitations of Related Work

To the best of our knowledge, this thesis is the first work aimed at: (i) carrying out a series of empirical studies on model composition effort so that a body of empirical knowledge in this field can be created and refined in the future; and (ii) defining support for the evaluation of model composition effort. In fact, it is well known that empirical studies in model composition are severely lacking. A previous roadmap study of model-driven software development (France & Rumpe, 2007) highlights that the state of the practice in assessing model composition provides evidence that the composition of design models is still in the “craftsmanship era.” In (Mens, 2002), the author also points out the need to empirically evaluate the effort that developers invest to compose software artifacts, in particular, when using the most commonly used design models, such as component diagrams and class diagrams.

This thesis identified two critical limitations in the current related work. First, the traditional measurement approaches are unable to support the analysis of model composition effort. Second, the current literature in model composition fails to provide empirical knowledge about how developers spend effort to produce an output intended model. These limitations are described as follows.

Limitation of Traditional Measurement Approaches

Researchers and developers are increasingly concerned with defining software metrics for different software engineering fields (Basili, 2007). This need is attested by the high number of many measurement approaches proposed over the last decade, e.g., (Chidamber & Kemerer, 1994; Fenton & Pfleeger, 1997; Chidamber et al., 1998). These measurement approaches focus on quantifying particular properties of software products. As far as evaluation of model composition effort is concerned, the conventional measurement approaches suffer from two types of major criticisms.

First, most of the existing product metrics is focused on supporting the assessment of particular forms of design decomposition, such as object-oriented (OO) software design. Typically, such metrics suites aim at quantifying attributes of OO systems, such as data abstraction, encapsulation, polymorphism, and inheritance usage. Such attributes often require more than one metric to be entirely characterized. Each metric quantifies properties of an object-oriented decomposition, such as classes and their relationships. The operational definition of these metrics relies on the constructs of the OO programming languages (e.g., Java and C++) and OO design modeling languages (e.g., UML). Examples of these constructs are UML packages, components, classes, and relationships that are specified in the UML metamodel.

For instance, Chidamber and Kemerer proposed a metrics suite to quantify some of these attributes in OO designs or programs (Chidamber & Kemerer, 1994). Examples of such metrics are coupling between objects, cohesion in methods, depth of inheritance, and so forth. In 1998, Chidamber and colleagues evaluated those metrics in order to assess their usefulness for practicing managers (Chidamber et al., 1998). In 1997, Fenton and Pfleeger formally analyzed the same metrics by applying basic criteria from measurement theory; their goal was also to offer an accessible and comprehensive introduction to software metrics with an emphasis on real-world applications (Fenton & Pfleeger, 1997). However, the aforementioned measurement approaches do not take into account the particularities of model composition activities. They only quantify static attributes of object-oriented software artefacts. Therefore, they cannot be directly used to improve our empirical understanding about model composition effort. These quantification methods are in stark contrast with the needs required by the effort measurement addressed in this thesis.

A second limitation of the existing measurement approaches is their inability to evaluate specific activities of model composition. During the composition process, developers execute a set of tasks to combine two input models (M_A and M_B) and produce an output intended model (M_{AB}). Examples of these tasks would be the application of the composition techniques and the resolution of inconsistencies in the composed model. The execution of each task consumes effort. By knowing the effort invested in each model composition task, developers may identify forms of alleviating the overall composition effort. Unfortunately, the traditional measurement approaches are unable to capture effort spent on specific model composition activities. Researchers do not know which and how model composition artefacts, produced in each task, should be quantified. This lack of effective measurement approaches for model composition effort also hinders the design and execution of empirical studies.

The Lack of Practical Knowledge on Model Composition Effort

Researchers and developers acknowledge the importance of practical knowledge about the model composition effort. In general, the current works propose new model composition techniques and superficially assess the proposed solutions. Reviewing the current literature, existing works make use of and evaluate software composition techniques in the realm of configuration management (Aiello, 2010a; Perry et al., 2001; Grinter, 1997; Rochkind, 1975). These studies focus on the composition of code and assess the technical feasibility of the techniques. Perry and colleagues investigated the composition of code in the context of collaborative software development (Perry et al., 2001). The authors realize an observational case study to understand how concurrent changes in large-scale software systems happen. The main results indicate that the degree of parallelism is very high, i.e., higher than considered by tools; and there is a significant correlation between the degree of parallel work on a given component and the number of quality problems it has.

However, little has been done to understand how developers invest effort in real-world settings. Today, it is well known that empirical studies on model composition are severely lacking. This scenario is still aggravated when considering composition effort. In fact, experts in the literature recently highlighted the scarcity of empirical studies (France & Rumpe, 2007). Additionally, the authors not only recognize but also recommend the execution of empirical studies to evaluate the impact of parallel changes on the development effort (Mens, 2002; Perry et al., 2001). In addition, they reinforce that empirical studies would allow researchers to evaluate the scalability of current composition techniques, to weigh the trade-offs in effort, and understand why and in what situations one approach might be better than another might.

In a broader context, we have also observed that many techniques have been proposed and incorporated into tools over the last decades. Examples of these techniques are SVN (SVN, 2011) and GIT (GIT, 2011). Using these tools, developers can control the evolution of software artefacts. In practice, these techniques help developers to check out artefacts for editing and then checking them back (Grinter, 1997; Rochkind, 1975). By controlling and registering these two activities, such techniques manage the evolution of the artefacts. In the seminar paper (Altmanninger et al., 2009), Altmanninger and colleagues apply the state-of-the-art versioning systems and analyze the challenges coming along with merging different versions of one model.

Other authors investigate the identification of conflicting changes by providing workspace awareness tools (Sarma et al., 2012; Burn et al., 2011a; Sarma et al., 2008). These tools are able to proactively identify overlapping changes between software artefacts such as code. The authors advocate that earlier contradicting changes are detected, the easier they are to resolve (Sarma et al., 2012). Sarma and colleagues propose a tool, named Palantír, which provides users with information about relevant ongoing parallel changes occurring in private workspaces, thereby enabling the early detection and resolution of potential conflicts.

Although these techniques are robust and broadly used in industry, nothing has been done to investigate about the effort to compose software artefacts. In (Uhl, 2008), Uhl points out that the model composition is more challenging than code composition. One of the reasons is because model composition involves the comparison and composition of graphical views, forms, dialogs, and property sheets as well as text. In fact, they are much more difficult to compare, mostly because visualizing the differences in a usable way is difficult. Moreover, Mens (Mens et al., 2002) also reinforces that the need for more empirical and experimental research regarding the amount of effort required resolving the composition inconsistencies.

To sum up, we observe that: (1) researchers do not even know which factors can, in fact, affect the composition effort; (2) nothing has been done to define how to evaluate the composition effort; and (3) there exists no cost effectiveness analysis about the model composition effort in order to support (or not) its well-informed use in practice.

1.3.
Study Methodology

The main goal of this thesis is to define an evaluation approach for model composition effort, thereby gathering empirical knowledge about the effort of composing design models. Based on this empirical knowledge, we aim at generating insight about how to reduce the composition effort model. This aimed will be achieved by understanding the side effects of influential factors on model composition effort. With this in mind, the goal of this study is formulated based on the GQM template (Basili et al., 1994) as follows:

General Goal: *Analyze* the influential factors *for the purpose of* investigating their effects *with respect to* model composition effort *from the perspective of* developers *in the context of* the evolution of design models.

To address that general goal, we formulate an overall research question (RQ), which is presented below:

- **$RQ_{overall}$:** How can the composition of design models be evaluated, in particular, with respect to developers' effort?

This general research question is elaborated into more detailed research questions, which require proper measurement means and empirical studies on model composition effort. The first research question (RQ1) addresses the need for providing an approach to support model composition evaluation. RQ1 is designed as follows:

- **RQ1:** How can the evaluation of model composition be organized in terms of a comprehensive framework?

The composition effort may be affected by a wide range of influential factors. In this thesis, we decided to study three factors that are fundamental to produce an expected output composed model: (i) the composition technique being employed, (ii) the design decomposition techniques, and (iii) the structural characteristics of the design models involved in the composition. The first factor is the type of model composition technique, which can be categorized into heuristic-based composition techniques (IBM RSA, 2011) and specification-based composition techniques (Epsilon, 2011). This factor, discussed in Section 2.4, may affect the effort that developers invest to combine the input models in order to produce an output intended model.

The second research question (RQ2) aims at evaluating the relative effort of composing the input models by applying heuristic-based and specification-based composition techniques. Each of these alternative techniques might require less effort in specific or all scenarios involving software evolution – the context of our studies of model composition. Then, we investigate the effects of using different composition techniques to produce the output intended model. RQ2 is stated as follows:

- **RQ2:** What is the effort of composing design models with specification-based composition techniques and heuristic-based composition techniques?

The third research question (RQ3) analyzes the effort of detecting inconsistencies. Detection of inconsistencies requires that developers inspect the elements of the composed model, which are structured according to the selected design decomposition. Therefore, we analyze the effects of significantly different forms of design decomposition (i.e., object-orientation and aspect-orientation) on the quality of the output composition. In particular, our goal is to understand how different design decompositions affect the inconsistency rate, the inconsistency detection effort, and the degree of misinterpretations of the output composed models. RQ3 is presented below:

- **RQ3:** What is the effect of design decomposition techniques in particular with respect to misinterpretation, inconsistency rate, inconsistency detection effort, and inconsistency resolution effort?

The fourth research question (RQ4) analyzes the effort of resolving inconsistencies. That is, we investigate the effort that developers invest to transform an output composed model into an intended model. Additionally, we analyze if well known design characteristics (Martin, 2003; Meyer, 1997), such as model stability (Section 2.6.1), may be used as an indicator of the presence of inconsistencies and of the effort to resolve inconsistencies. RQ4 is stated as follows.

- **RQ4:** What is the impact of design characteristics on the inconsistency rate and inconsistency resolution effort?

Our studies to answer these research questions are viewed as the key original contribution of this work. No previous work has studied these different dimensions of model composition effort until now. It is important to highlight that we aim at investigating these research questions in the context of composing well-known design models, including UML class diagrams and architectural models, which are the most used design models in practice (Dobing & Parsons, 2006). While we mostly focus on structural design models in our studies, behavioral models were also involved in one of the studies. The next section discusses the thesis contributions more carefully.

1.4. Thesis Contributions

The previous sections discussed the limitations of related work, stated the research problem being addressed, and then presented the study methodology. This section describes the thesis contributions, which consist of an evaluation approach and the production of empirical knowledge about model composition effort. All contributions are derived from a series of empirical studies, including controlled experiments, quasi-experiments, case studies, interviews, and observational studies. These qualitative and quantitative studies evaluate the composition effort from different perspectives in realistic and controlled contexts by collecting multiple sources of evidence. More specifically, the contributions of this thesis are the following:

1. *A quality model for model composition effort (RQ1).* Some quality models for design modeling have been previously proposed. Some examples are described in (Lange, 2007a; Krogstie, 1995; Lindland et al., 1994). However, these quality models aim at software modeling in general rather than model composition effort. The contribution of this thesis is, therefore, the extension of the existing quality models for model composition effort. The extension is based on practical knowledge derived from our experience in conducting a range of empirical studies, including two controlled experiments, five industrial case studies, three quasi-experiments, interviews, and seven observational studies. Therefore, our evidence-based quality model provides guidance to developers and researchers about how to plan empirical studies in model composition. The guidance is characterized by: (i) a unifying terminology for activities and artefacts involved in model composition tasks, and (ii) the systematic relation between quality notions and metrics for the qualitative and quantitative assessment in the realm of model composition.

These elements of the quality model can also help to identify and empirically evaluate possible factors or indicators of model composition effort. For instance, the quality model helped us to select metrics and procedures to evaluate how the three influential factors (i.e., design decompositions, the design characteristics, and the composition techniques) affect model composition. The quality model can also serve as a reference frame to structure empirical studies performed by other researchers in the future. Without a reference frame, the replication and comparison of empirical studies as well as the generalization of their results are jeopardized. Chapter 3 elaborates the quality model.

2. *Insight and practical knowledge on model composition effort (RQ2-4).* The quality model guides the investigation about the effects of factors on the model composition effort. As previously mentioned, three factors are considered in this thesis: (1) the composition techniques (Section 2.4), (2) the design modeling technique used to decompose the design models (Section 2.5), and (3) the model stability (Section 2.6). The evaluation is performed by a series of experimental studies including: two controlled experiments, five industrial case studies, three quasi-experiments, more than fifty interviews, and seven observational studies. The empirical findings enhance the knowledge about the impact of the influential factors on: (i) the effort to apply model composition techniques; (ii) effort to detect inconsistencies; and (iii) the effort to resolve inconsistencies. Additionally, we gather insight about how to evaluate the developers' effort, reduce error proneness in model composition, and tame side effects of the influential factors in practice. The current body of knowledge on model composition is improved as our studies allowed to: (i) test out recurring claims, which were formulated by the experts in the literature, but that were never evaluated; (ii) identify correlations between key dependent and independent variables involved in model composition; for instance, identify which types of changes make model composition an error-prone and effort-consuming task; (iii) build a clear understanding to further support the formulation of theories on model composition; (iv) provide a solid background to inspire the creation of the next-generation model composition techniques and tools; and (v) pinpoint when the model composition techniques work and when they do not work.

These contributions are presented and discussed throughout the next chapters, and refined in Chapter 7. They have been reported in a number of papers, where part of them were already published in international conferences and workshops or submitted to journals. Table 1 shows the list of publications that are related to the thesis directly and indirectly.

1.5. Thesis Outline

This section outlines how the contributions are reported in each chapter, and makes explicit the relation between the chapters and the research questions.

Chapter 2: Background and Related Work. It defines the main concepts used throughout this thesis. These definitions are essential to understand the contributions and the results achieved. In addition, this chapter discusses related work, contrasting the commonalities and differences with respect to our research.

Chapter 3: A Quality Model for Model Composition (RQ1). This chapter sets up the context for proposing a quality model for model composition effort by discussing the limitations of existing quality models. After that, the chapter introduces the quality model, which provides the basis for all empirical studies realized throughout this research. This quality model takes into account the elements relevant to the three influential factors investigated in our empirical studies: the model composition techniques (Section 2.4), the design modeling languages (Section 2.5), and the design characteristics (e.g., model stability) (Section 2.6). More specifically, the quality model relates composition metrics and a series of quality notions, such as semantic, syntactic, and social quality notions. The quality model also serves as a practical guideline to select metrics and procedures to evaluate how the influential factors affect the model composition. This chapter elaborates on initial ideas reported in (Farias et al., 2008a).

Chapter 4: Effort on the Application of Composition Techniques (RQ2). This chapter reports upon the effects of composition techniques — both specification-based techniques and heuristic ones — on the developers' effort and its relation to the correctness of the output composed models. This cost-effectiveness analysis of the techniques is realized based on a range of quantitative and qualitative empirical studies including one controlled experiment, five industrial case studies, observational studies, and interviews. These combined studies allow building a body of knowledge about the effort that developers invest to compose design models. It is expected that the specification-based techniques reduce the developers' effort and assure the correctness of the compositions when compared to the heuristic-based techniques. However, the results, supported by a comprehensive set of statistical analyses, reveal the opposite, the specification-based techniques increase the developers' effort and do not assure the correctness of the compositions when compared to the heuristic-based techniques. The results presented in this chapter are presented in three papers (Farias, 2011a; Farias et al., 2012a; Farias et al., 2012c).

Chapter 5: Effort on the Detection of Inconsistencies (RQ3). This chapter investigates the effects of significantly different forms of design decomposition (i.e., object-oriented modeling and aspect-oriented modeling) on the effort to detect inconsistencies in the output composed model. The results provide insight about the impacts of using different modeling languages on the effort of detecting inconsistencies. As in the previous studies, this insight is generated from a family of experimental investigations including one controlled experiment, five industrial case studies, observational studies, and interviews. These studies allowed investigating RQ3 from different perspectives, i.e., varying the artifacts analyzed, the context (in vivo and in vitro), and the cultural biases in applying the evaluation (companies and university in different locations). Elements of this chapter were reported in three papers (Farias et al., 2012b; Farias, 2011a; Medeiros et al., 2010).

Chapter 6: Effort on the Resolution of Inconsistencies (RQ4). This chapter investigates the effort that developers spend to resolve inconsistencies. In particular, we study the influence of modeling languages and model stability on the production of inconsistencies and on the effort to resolve these inconsistencies. As in the previous chapter, the findings and lessons learned are gathered from a multiple studies, including two quasi-experiments in the context of evolving design models. All results are supported by statistical tests. Elements of this chapter are reported in papers as well (Farias et al., 2012d; Farias et al., 2010a; Farias et al., 2010b; Farias et al., 2011).

Chapter 7: Conclusions. This chapter presents a summary of our research, a refinement of the contributions, and the final remarks.

Direct Publications	RQ
1. Kleinner Farias, Alessandro Garcia, and Carlos Lucena, *Evaluating the Impact of Aspects on Inconsistency Detection Effort: A Controlled Experiment*. In: 15th International Conference on Model-Driven Engineering Languages and Systems (MODELS), Foundations Track, Austria, 2012.	RQ3
2. Kleinner Farias, Alessandro Garcia, Jon Whittle, Christina Chavez, and Carlos Lucena, *Evaluating the Effort of Composing Design Models: A Controlled Experiment*, In: 15th International Conference on Model-Driven Engineering Languages and Systems (MODELS), Applications Track, Austria, 2012.	RQ2
3. Kleinner Farias, Alessandro Garcia and Jon Whittle, *Assessing the Impact of Aspects on Model Composition Effort*, In: 9th International Conference on Aspect-Oriented Software Development (AOSD'10), Saint-Malo, France, 2010 (Indicated to Best Paper Award - Accept. Rate < 30%).	RQ3, RQ4
4. Kleinner Farias, Alessandro Garcia, Carlos Lucena, *Evaluating the Effects of Stability on Model Composition Effort: an Exploratory Study*, Journal of Software and Systems Modeling, 2012.	RQ4
5. Kleinner Farias, Alessandro Garcia, Jon Whittle, and Carlos Lucena, *Analyzing the Effort on Composing Design Models of Large-Scale Software*, IEEE Transactions on Software Engineering, 2012. (Submitted)	RQ2
6. Kleinner Farias, *Empirical Evaluation of Effort on Composing Design Model*, In: Doctoral Symposium at the International Conference on Software Engineering (ICSE'10), pages 405-408, South Africa, 2010.	All
7. Kleinner Farias, Alessandro Garcia and Jon Whittle, *On the Quantitative Assessment of Class Model Compositions: An Exploratory Study*, In: Empirical Studies of Model-Driven Engineering (ESMDE'08) at MODELS'08, v. 1, pages 1-10, 2008.	all
8. Kleinner Farias, Alessandro Garcia, Carlos Lucena, *Evaluating the Effects of Stability on Model Composition Effort: an Exploratory Study*, In: VIII Experimental Software Engineering Latin American Workshop at XIV Iberoamerican Conference on Software Engineering, April, Rio de Janeiro, pages 81-91, 2011.	RQ4
9. Kleinner Farias, *Analyzing the Effort on Composing Design Models in Industrial Case Studies*, In: 10th International Conference on Aspect-Oriented Software Development Companion, pages 79-80, Porto de Galinhas, Brazil, 2011.	all
10. Ana Luisa Medeiros, Kleinner Farias, Alessandro Garcia, and Thais Batista, *Evaluating Composition Techniques for Architectural Specifications: A Comparative Study*, In: Empirical Evaluation of Software Composition Techniques (ESCOT 2010) at AOSD'10, Rennes, France, 2010.	RQ2, RQ3
11. Everton Guimarães, Alessandro Garcia, and Kleinner Farias, *Analyzing the Effects of Aspect Properties on Model Composition Effort: A Replicated Study*, In: 6th Workshop on Aspect-Oriented Modeling at MODELS'10, Oslo 2010.	RQ2, RQ3
12. Kleinner Farias, Alessandro Garcia and Carlos Lucena, *On the Comparative Evaluation of Aspect-Oriented Model Composition Techniques*, In: III Latin-American Workshop on Aspect-Oriented Software Development (LA-WASP´09) at XXIII Brazilian Symposium on Software Engineering, pages 45-49, Ceará, Brazil, 2009.	all

Indirect Publications
1. Kleinner Oliveira, Karin Breitman, Toacy Oliveira, *A Flexible Strategy-Based Model Comparison Approach: Bridging the Syntactic and Semantic Gap*, Journal of Universal Computer Science, v. 15, p. 2225-2253, 2009.
2. Kleinner Farias, Ingrid Nunes, Viviane Silva, Carlos Lucena, *MAS-ML Tool: Um Ambiente de Modelagem de Sistemas Multi-Agentes*, In: Workshop on Software Engineering for Agent-oriented Systems at XXIII Brazilian Symposium on Software Engineering, Ceará, Brazil, 2009
3. Enyo Gonçalves, Kleinner Farias, Mariela Cortes, Viviane Silva, Ricardo Feitosa, *Modelagem de Organizações de Agentes Inteligentes: uma Extensão da MAS-ML Tool*, In: 1st Workshop on Autonomous Software Systems, CBSoft 10, 2010, Salvador, Bahia, 2010.
4. Enyo Goncalves, Kleinner Farias, Mariela Cortes, Alexandre Feijo, Fabiano Oliveira, Viviane Silva, *MAS-ML Tool: A Modeling Environment for Multi-Agent Systems*, In: 13th International Conference on Enterprise Information Systems (ICEIS), 2011, Beijing, China 2011.
5. Kleinner Oliveira, Karin Breitman, Toacy Oliveira, *Ontology Aided Model Comparison*, In: Fourteenth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS`09), p. 78-83, Potsdam, Germany, 2009.

Table 1: List of direct and indirect publications