Assessing Composition in Modeling Approaches

Gunter Mussbacher[1], Omar Alam[2], Mohammad Alhaj[1], Shaukat Ali[3], Nuno Amálio[4], Balbir Barn[5], Rolv Bræk[6], Tony Clark[5], Benoit Combemale[7], Luiz Marcio Cysneiros[8], Urooj Fatima[6], Robert France[9], Geri Georg[9], Jennifer Horkoff[10], Jörg Kienzle[2], Julio Cesar Leite[11], Timothy C. Lethbridge[12], Markus Luckey[13], Ana Moreira[14], Felix Mutz[13], A. Padua A. Oliveira[15], Dorina C. Petriu[1], Matthias Schöttle[2], Lucy Troup[16], Vera M.B. Werneck[15]

[1] SCE, Carleton University, Ottawa, Canada
{gunter | malhaj | petriu}@sce.carleton.ca

[2] SCS, McGill University, Montreal, Canada
Omar.Alam@mail.mcgill.ca, Joerg.Kienzle@mcgill.ca, mschoettle@cs.mcgill.ca

[3] Certus Software V&V Center, Simula Research Laboratory, Norway
shaukat@simula.no

[4] LASSY, University of Luxembourg, Luxembourg
nuno.amalio@uni.lu

[5] SEIS, Middlesex University, London, UK
{b.barn | t.n.clark}@mdx.ac.uk

[6] Dept. of T., The Norwegian Uni. of Science and Technology, Norway
{rolv.braek | urooj}@item.ntnu.no

[7] IRISA, Rennes, France
benoit.combemale@irisa.fr

[8] York University, Toronto, Canada
cysneiro@yorku.ca

[9] CS Dept., Colorado State University, Fort Collins, USA
{france | georg}@cs.colostate.edu

[10] DISI, University of Trento, Trento, Italy
horkoff@disi.unitn.it

[11] Pontifícia Universidade Católica do Rio de Janeiro (PUC-RIO), Brazil
julio@inf.puc-rio.br

[12] EECS, University of Ottawa, Ottawa, Canada
tcl@eecs.uottawa.ca

[13] University of Paderborn, Paderborn, Germany
{mluckey | fmutz}@mail.upb.de

[14] Universidade Nova de Lisboa, Lisbon, Portugal
amm@fct.unl.pt

[15] Universidade do Estado do Rio de Janeiro (UERJ), Brazil
padua.uerj@gmail.com, vera@ime.uerj.br

[16] Dept. of Psychology, Colorado State University, Fort Collins, USA
Lucy.Troup@colostate.edu

ABSTRACT
Modeling approaches are based on various paradigms, e.g., aspect-oriented, feature-oriented, object-oriented, and logic-based. Modeling approaches may cover requirements models to low-level design models, are developed for various purposes, use various means of composition, and thus are difficult to compare. However, such comparisons are critical to help practitioners know under which conditions approaches are most applicable, and how they might be successfully generalized and combined to achieve end-to-end methods. This paper reports on work done at the 2nd International Comparing Modeling Approaches (CMA) workshop towards the goal of identifying potential comprehensive modeling methodologies with a particular emphasis on composition: (i) an improved set of comparison criteria; (ii) 19 assessments of modeling approaches based on the comparison criteria and a common, focused case study.

CMA'12, September 30 2012, Innsbruck, Austria

Categories and Subject Descriptors
D.2.1 [**Software Engineering**]: Requirements/Specifications – *Languages.* D.2.2 [**Software Engineering**]: Design Tools and Techniques.

General Terms
Documentation, Languages.

Keywords
Composition, Modeling, Comparison Criteria, Case Study.

1. INTRODUCTION

There are many modeling approaches covering requirements to low-level design that support different paradigms such as object-oriented, aspect-oriented, procedural, service-oriented, goal-oriented, component-based, feature-oriented, workflow/scenario-based, agent-oriented, and logic-based paradigms. Different approaches also offer a wide range of composition rules and operators, making it difficult to compare them and know under which conditions different modeling approaches are most applicable. However, it is crucially important to compare modeling approaches in order to integrate existing modeling approaches and to generalize individual approaches into a comprehensive end-to-end method. Such a method that spans from early requirements to low-level design and provides well-defined composition rules and operators across the whole software development cycle does not yet exist, and it is not readily evident how such a method would actually work in practice. As part of identifying potential comprehensive methodologies, we must be able to compare different modeling approaches with each other.

The Second International Comparing Modeling Approaches (CMA) workshop at Models 2012 builds on the results of the two AOM Bellairs workshops in 2011 [1] and 2012 [2] as well as the inaugural CMA workshop at Models 2011 [3]. At these workshops, the focused bCMS case study [4] based on the original Crisis Management System (CMS) case study [5] was developed and a collection of comparison criteria for modeling approaches was initiated and further refined.

The workshops in 2011 provided the groundwork for a comprehensive comparison criteria document, covering software development phases and activities of a modeling approach, its relationship to standards, its semantics, modularity and composability issues, traceability and tool support issues, but also identifying several candidate comparison criteria to be included in future versions of the document. In the 2012 workshops, these initial criteria are further consolidated and described more formally with the help of a metamodel [6]. Furthermore, the modularity and composability sections in the comparison criteria document are significantly improved. In particular, we see composition as the act of building larger pieces from smaller ones; in the context of modeling, we take composition to mean the act of creating new first class entities of the modeling approach from existing ones (e.g., by putting together several units of encapsulation). This definition allows us to define a taxonomy of composition operations, from creating relations between existing model elements to merging multiple models without creating new elements that do not exist in the original models, and ranging from scopes of large modeling units to portions of modeling elements.

The comparison criteria document is fundamentally important because common terminology tends to be interpreted differently depending on someone's modeling background, requiring further clarifications and examples in the comparison criteria document. The term composition is a prototypical example. In the context of comparing modeling approaches from various paradigms, composition needs to be defined rather broadly as will be discussed in this paper.

The first CMA workshop was mostly submitter-driven and resulted in the assessment of six modeling approaches [7]. The focused bCMS case study was crucial in this effort as it allowed the *entire* case study to be modeled, hence providing a solid basis for discussion, comparison, and evaluation while still making it possible to demonstrate the capabilities of a modeling approach. Therefore, the second CMA workshop continues to use the bCMS case study. Before holding the workshop, the authors of a modeling approach apply their approach to the bCMS case study and assess their approach with the help of the comparison criteria. These assessments are contrasted and discussed at the workshop, leading to the refinement and correction of comparison criteria during the workshop. The discussions during the workshop focus mostly on the comparison criteria for composability. The improved comparison criteria are applied again to the modeling approaches after the workshop.

The 2012 edition of CMA takes a different approach than the first edition in that specific modeling approaches are actively solicited on a much larger scale [8], resulting in 19 assessments of 14 modeling approaches which are discussed in this paper. All assessments are available on the CMA 2012 workshop page in the Repository for Model-Driven Development (ReMoDD) [9]. The resulting 14 models of the bCMS case study of the various modeling approaches are also available in ReMoDD (11 new or updated, 3 from CMA 2011). The difference in numbers (19 assessments vs. 14 modeling approaches vs. 14 bCMS models) is due to the fact that bCMS models are not available for some modeling approaches (and fortunately are not needed to perform the assessment) and that the assessment of the UML-based modeling approach yields individual assessments of six notations.

The collected assessments are the first steps towards the ability to search for a language with specific characteristics, or for a person building a language to understand what already exists. Furthermore, this year's focus on composability contributes to an emerging taxonomy of composition specifications, i.e., a set of language operators and rules that can be used to compose various models.

The remainder of this paper gives a definition of composition as required for our context in Section 2. Section 3 briefly introduces the covered modeling approaches. Section 4 presents initial analysis results from the 19 assessments while Section 5 concludes the paper and discusses future work. The appendix presents relevant excerpts from the assessments.

2. DEFINITION OF COMPOSITION

In our context, composition or *composability* is defined rather broadly to capture the different notions and interpretations of composition relevant to a wide range of very different modeling approaches. Some may view composition as an operation that is performed on larger modeling units (e.g., a security concern is composed with a performance concern) but not at finer levels of granularity. Some may view composition as the act of merging two models together without adding any new model elements that do not exist in either of the two source models (e.g., security-specific methods are merged with existing classes to provide security-related behavior for a system). Others may view composition as establishing some kind of relationship or link between modeling elements (e.g., adding an association between two classes or adding a number of contribution links between the goal model of the security concern and the goal model of the system).

We define composition as the act of creating new first class entities of the modeling approach from existing ones (e.g., by putting

Legend: a black square (■) indicates that the phase or activity applies to the modeling approach.

	Adapt Case	AoURN	AspectSM	AT	Int. RE	i*	Kermeta	LEAP	MDSE	PUMA4SOA	RAM	UML	Umple	VCL
Early requirements		■		■	■	■		■						
Late requirements	■	■						■	■			■	■	■
Architecture/High-level design	■	■			■	■	■	■	■	■	■	■	■	■
Detailed/Low-level design	■		■				■		■	■	■	■	■	■
Implementation							■		■		■	■	■	
Integration												■		
Deployment												■		
Specification/Modeling	■	■		■	■	■	■	■	■	■	■	■	■	■
Validation		■	■		■	■	■	■	■			■		■
Verification	■	■			■	■	■	■	■	■		■	■	■
Evolution		■		■		■			■			■	■	
Analysis		■		■	■	■			■	■		■		■
Trade-off Analysis		■				■				■				

Figure 1. Software Development Phases and Activities of Modeling Approaches

together several units of encapsulation). The main defining characteristic is hence that composition combines existing first class entities in some way. A composition is specified either as a *composition rule* or a *composition operator*, is applied to some input model elements, and produces some output. A composition rule provides the specification of a composition but does not actually perform the composition (e.g., a binding rule specifies that two model elements are to be merged). A composition operator, on the other hand, results in a composed model (e.g., a merge operator actually merges the two model elements into one).

Some composition rules and composition operators enable the structuring of modules through traditional means. These include association, generalization (inheritance), aggregation, and composition (as defined by UML), hierarchical decomposition (e.g., sub-activity diagrams), grouping mechanisms (e.g., a package), as well as containment mechanisms (e.g., an activity diagram containing activity nodes). The latter three are very common forms of composition for many modeling approaches. More advanced forms of composition rules and composition operators are used for (i) the composition of crosscutting concerns through pattern matching, superimposition, aspect weaving, or other means and (ii) model transformations.

3. MODELING APPROACHES

The following modeling approaches, listed in alphabetical order, are the basis for the 19 assessments given the comparison criteria. The references for each modeling approach constitute the resources used for the assessments.

Activity Theory (AT)
The AT approach addresses the issue of highly diverse stakeholders, or situations where not all stakeholders may be known, and the lack of a common set of goals across stakeholders.
Resources: [10], [11], [12]

Adapt Case
The Adapt Case approach captures structural and behavioral adaptation in software system models for high-level and low-level design, by providing a middle-weight extension to UML.
Resources: [13], [14], [15]

Aspect-oriented User Requirements Notation (AoURN)
AoURN supports requirements engineering activities from elicitation to specification and analysis to validation in the presence of crosscutting concerns in goal and scenario models.
Resources: [16], [17], [18], [19], [20]

AspectSM
AspectSM's purpose is to model robustness behavior on UML state machines for robustness test case generation while taking crosscutting behavior into account.
Resources: [21], [22], [23], [24], [25], [26], [27], [28]

Intentional Requirements Engineering
Intentional Requirements Engineering aims to fill a method gap in the i* framework by providing engineering-driven systematic steps towards the elaboration of modular i* models.
Resources: [29], [30], [31], [32], [33], [34], [35], [36]

i*
i* captures the intentional and social aspects of early requirements analysis, allowing modelers to explore alternative requirements, trade-offs, and the "why" behind requirements.
Resources: [29], [37], [38], [39]

Kermeta
Kermeta is mainly used at the language level to design and implement domain-specific modeling languages and their respective tools (transformations, compositions, simulators, compilers…).
Resources: [40], [41], [42], [43]

LEAP
LEAP captures system architecture requirements, models the architecture to the level of operational components, aligns requirements with architecture, and provides simulations.
Resources: [44], [45], [46], [47]

Model Driven Service Engineering (MDSE)
MDSE addresses high-level and compositional service specification allowing for complete service behavior definitions and semi-automatic design synthesis as well as realizability analysis. Resources: [48], [49], [50], [51]

Performance from Unified Modeling Analysis for SOA (PUMA4SOA)
PUMA4SOA derives performance models from UML design models of SOA enterprise systems to evaluate their run-time performance from early development phases.
Resources: [52], [53], [54], [55], [56]

Reusable Aspect Models (RAM)
RAM is a reuse-oriented, multi-view modeling approach targeted at high-level and low-level software design with aspect-oriented modeling techniques for class, sequence, and state diagrams. Resources: [57], [58], [59], [60], [61]

Unified Modeling Language (UML)
UML's activity diagrams, class diagrams, component diagrams, sequence diagrams, state machines, and use case diagrams are assessed.
Resources: [50]

Umple
Umple seeks to bring modeling abstractions directly into textual programming languages and provides a model-editing environment with code generation as good as any compiler.
Resources: [62], [63], [64]

Visual Contract Language (VCL)
VCL is a language to model software designs visually and formally based on set theory and design-by-contract (pre/post-conditions), while abstracting away several implementation details. Resources: [65], [66], [67], [68]

The above modeling approaches cover all software development phases from early requirements to implementation and to a limited extend also integration and deployment as shown in Figure 1. The modeling approaches are applicable to the software development activities of specification/modeling, validation, verification, evolution, analysis, and trade-off analysis as depicted in Figure 1.

All of the modeling approaches are considered general purpose, and hence applicable not only for a specific domain but all domains, with the exception of AspectSM, AT, LEAP, and arguably PUMA4SOA.

AspectSM is best suited to the specific application domains of communication and control systems as well as embedded and real-time systems. AT, on the other hand, is preferably applied to cyber-physical systems where the humans-in-the-loop or key stakeholder groups may not share the same end goals, but it should not be applied to well-understood systems if stakeholders agree on end goals as AT requires considerable effort in terms of time. LEAP should not be applied to real-time systems but rather to information systems and enterprise architectures. Finally, PUMA4SOA is basically a general-purpose language but focuses on scenario-based performance analysis.

4. RESULTS OF THE ASSESSMENTS

In addition to the phases and activities shown in Figure 1, we present three major results of our analysis of the assessments in this section. The first is distinct relations and groupings among the approaches. The second relates to the paradigms embodied by the approaches and the relative formality of the various approaches, and the third relates to their use of composition rules and operators.

The modeling approaches covered by the CMA workshops are not isolated from each other but relate to each other as shown in Figure 2. Two rather distinct groups can be observed in this figure: (i) at the top of the figure, the modeling approaches focusing mainly on requirements (AoURN, AT, Intentional RE, and i*) as well as LEAP (which deals with requirements but does also significantly focus on downstream activities) and (ii) at the bottom

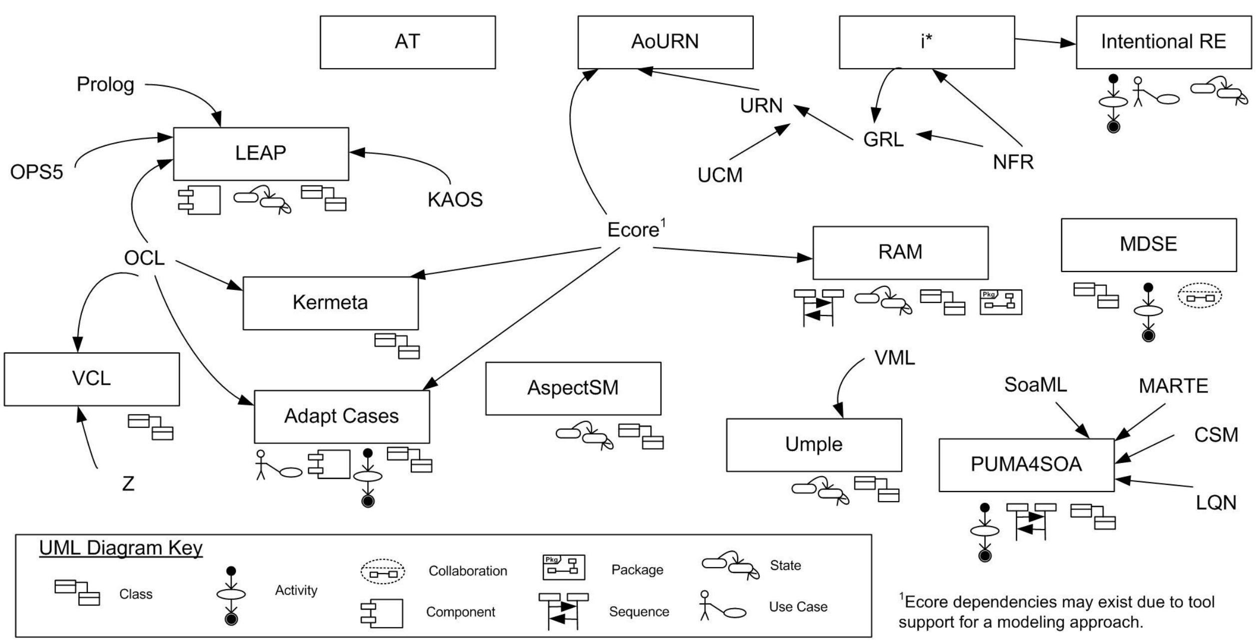

Figure 2. Relationships of CMA Workshop Modeling Approaches

Legend: a black square (■) indicates that the paradigm or level of formality applies to the modeling approach.	Adapt Case	AoURN	AspectSM	AT	Int. RE	i*	Kermeta	LEAP	MDSE	PUMA4SOA	RAM	UML	Umple	VCL
Object-oriented	■		■				■	■	■	■	■	■	■	■
Aspect-oriented		■	■							■	■		■	■
Procedural		■						■				■		
Service-oriented		■						■	■	■		■		
Workflow/scenario-based		■								■				
Goal-oriented		■		■	■	■		■						
Agent-based					■	■								
Component-based	■							■	■	■		■	■	
Feature-oriented		■							■	■	■		■	
Logic-based								■						■
Formal	■					■				■				■
Rigorous		■	■		■	■	■	■	■	■	■	■	■	
Informal				■										

Figure 3. Paradigms and Semantics of Modeling Approaches

of the figure, the remaining modeling approaches which emphasize later software development phases.

Not surprisingly, a similar grouping can also be witnessed for the paradigms of the modeling approaches. In total, the modeling approaches are influenced by ten paradigms as illustrated in Figure 3. All modeling approaches except those focusing mainly on requirements are object-oriented while all requirements modeling approaches (AoURN, AT, Intentional RE, and i*) and LEAP are goal-oriented. Almost half of the modeling approaches are aspect-oriented (AoURN, AspectSM, PUMA4SOA, RAM, Umple, and VCL). Component-based (Adapt Case, LEAP, MDSE, PUMA4SOA, UML, and Umple) and feature-oriented modeling approaches (AoURN, MDSE, PUMA4SOA, RAM, and Umple) are also well represented. Only two requirements modeling approaches are agent-based (Intentional RE and i*), two other modeling approaches are workflow/scenario-based (AoURN and PUMA4SOA), while yet two other modeling approaches are logic-based (LEAP and VCL).

The approaches also vary in the extent to which they can be considered formal. This characteristic is also shown in Figure 3. The second logic-based modeling approach (VCL) and Adapt Cases are the only ones that are classified exclusively as an approach with formal semantics, i.e., the language is based upon a formal domain that is mathematically well-understood and that allows proofs to be performed or it is possible to map the language to mathematical logic expressions. A language with formal semantics is entirely expressed in mathematical terms. Such a language is mechanically and exhaustively analyzable, which means that a machine can be used to check properties of models, using theorem proving or model checking, in a way that all states of the modeled system are covered and the analysis gives an absolute guarantee on whether the model satisfies the property or not. However, there are theoretical and practical limitations on what can be analyzed formally and exhaustively; often such approaches do not scale well. A language with a rigorous semantics is a language with semantics expressed in a form that allows language statements to be mechanically analyzed in a limited way (i.e., not all aspects of the language are formalized – just enough to perform the types of analysis needed). Executable modeling languages fall into this category as they can be used to support simulations and testing, but one cannot use them to prove that all behaviors satisfy certain properties. Often, languages with a rigorous semantics are based on a well-defined metamodel. While this is usually sufficient to qualify as being rigorous, a well-defined metamodel by itself is insufficient to qualify as being formal. Finally, a language that is informal has none of the above characteristics (in particular, it is not machine analyzable). Consequently, all other modeling approaches are deemed to be rigorous with the exception of AT which is classified as informal at this point in time. This classification, however, may be upgraded to rigorous in the near future as work is underway to make AT amendable for machine-analysis. Finally, a part of PUMA4SOA is formal because PUMA4SOA uses Layered Queuing Networks (LQN) to assess performance and i* can also considered to be formal but several not-commonly agreed formalizations exist at this point.

We also explore the use of composition rules and operators in the different approaches. An excerpt of the most relevant assessment results related to composability is shown in the appendix for all modeling approaches. Some approaches combine composition rules and operators, others define operators, and others define rules.

As an example for a combination of composition rule and operator, consider *UML Generalization* which is applied to two classes. *UML Generalization* is a composition rule as it defines which two classes from one or two models should be used to establish generalization between them but the composition is not actually performed at the time of specification. The signature of the composition rule is $M \times C_1 \times M \times C_2 \rightarrow M'$ (where M are models that may or may not be the same, C_1 is the parent class, and C_2 is the child class that cannot be the same as C_1; the result is a new model M' with a new model element for generalization added). Any analysis of the model has to take this new model element into account.

However, there also exists a *UML Generalization* composition operator that is synonymous with the composition rule and performs the actual composition (i.e., adding elements and constraints to the classes so that the generalization composition rule is respected). Its signature is hence M x C_1 x C_2 → M' (where C_1 is the parent class, C_2 is the child class that cannot be the same as C_1, and both are connected in the model M by a generalization as specified by the composition rule; the result is a new model M'). Just as the composition rule introduces a new model element (i.e., the generalization), the composition operator also introduces new model elements that are necessary to express the semantics of the generalization composition rule. Since the composition operator takes the composition rule and applies it, the rule no longer is needed (i.e., it is not in the resulting model). Because there exists a dedicated composition operator used only for the composition rule *UML Generalization*, *UML Generalization* is categorized as both a rule and an operator.

An example of only a composition operator is *Control Flow Construct* in a scenario notation such as Use Case Maps (UCM) in AoURN which combines the building blocks of the scenario notation (i.e., map elements in UCM) in, e.g., sequence, parallel, or as alternatives. *Control Flow Construct* is only a composition operator as the act of specifying the operator immediately results in the composition being performed. The inputs of the *Control Flow Construct* are two map elements from one or two UCM models. The operator results in an updated UCM model where the map elements have been connected. Hence, the signature of the *Control Flow Construct* is M x Map $Element_1$ x M x Map $Element_2$ → M'. *Control Flow Construct* does add new model elements, i.e., those representing the control flow construct. Note that *Control Flow Construct* is not a composition rule with a synonymous composition operator like *UML Generalization*, because generalization can be expressed in UML with already existing language concepts such as class attributes and constraints while the control flow construct cannot be expressed in UCM with other existing language concepts.

Finally as an example of compositions that are only rules, the RAM approach uses *Instantiation (Customization)* which is a composition rule as it only defines which model elements from two source models need to be composed together. Its signature is $Aspect_1$ x Model $Element_1$ x $Aspect_2$ x Model $Element_2$ → $Aspect'_1$. *Instantiation (Customization)* is not a composition operator, as the composition operator used to realize *Instantiation (Customization)* is the composition operator *Class Merge* and this operator is not specific to *Instantiation (Customization)*. *Class Merge*, on the other hand, is only a composition operator with the signature $Class_1$ x $Class_2$ → $Class'_1$.

Figure 4 gives an overview of the assessment results of the 14 modeling approaches with respect to composability. The first row of the table indicates the total number of composition rules, operators, or combined rule/operators in the modeling approach. These three categories are from here on referred to collectively as composition rule/operators. The numbers of each category are given in the next lines of the table. The rest of the table rows are given as percentages (e.g., 80 for "Introduces new elements" for AoURN means that 80% of AoURN composition rules/operators introduce new elements whereas 20% do not).

As expected, the modeling approaches cover both composition rules and operators. Eight modeling approaches, however, support either composition rules (AspectSM, AT, Intentional RE, i*) or composition operators (Kermeta, LEAP, MDSE, PUMA4SOA) but not both. A majority of the modeling approach composition rules/operators do not introduce new modeling elements. In fact, only 33% of the composition rules/operators do introduce new elements, and half of the modeling approaches do not have any composition rule/operator that introduces new modeling elements.

	Adapt Case	AoURN	AspectSM	AT	Int. RE	i*	Kermeta	LEAP	MDSE	PUMA4SOA	RAM	UML	Umple	VCL	Total
Number of composition rules/operators	2	10	2*	5	2	2	3	5	4*	3	6	13*	6	10	73*
Composition rules	1	2	1	5	2	2	0	0	0	0	5	4	0	3	25
Composition operators	1	8	0	0	0	0	3	5	1	3	1	1	4	7	34
Combined rules and operators	0	0	0	0	0	0	0	0	0	0	0	4	2	0	6
Introduces new elements (%)	0	80	0	0	100	50	0	0	75	67	0	31	67	0	33
Input identification**															
Explicit input identification (%)	100	100	100	100	100	100	67	100	100	33[1]	100	100	100	100	96
Pattern matching (%)	0	40	0	0	0	0	33	0	0	0	67	0	17	0	14
Bindings (%)	0	0	0	0	0	0	0	0	75	33	50	0	0	0	10
Explicit application (%)	100	80	100	100	100	100	100	100	100	100	83	100	83	100	95
Symmetric (%)	0	0	0	0	0	0	67	0	100	67	33	31	0	100	33
Semantics-based (%)	100	20	0	0	100	100	0	40	0	100	17	15	17	0	23
Deterministic (%)	100	100	100	100	n/s	100	100	100	100	100	100	100	100	100	100

*) some categorized neither as rule nor as operator - see appendix for details
**) the subcategories of input identification may not sum up to 100% as some composition rules/operators may be categorized more than once
1) 33% not specified
n/s = not specified

Figure 4. Composability of Modeling Approaches

Most modeling approaches define composition rules/operators where the inputs to the composition rule/operator are explicitly identified (96%). Almost half of the modeling approaches, however, also support pattern matching (AoURN, Kermeta, RAM, Umple) or bindings (MDSE, PUMA4SOA, RAM). However, the overall number of composition rules/operators that support pattern matching and bindings is low (14% and 10%, respectively). Only three modeling approaches (AoURN, RAM, Umple) feature composition operators that are applied implicitly, i.e., a default composition mechanism is used (aspect marker insertion for AoURN, class merge for RAM, and mixin for Umple).

Only few modeling approaches use symmetric composition rules/operators (33%), while the majority employs asymmetric composition rules/operators. A composition rule/operator is typically applied to two or more input models. If all inputs are of the same type, then it is possible that symmetric composition is supported, i.e., the order of the inputs does not matter (e.g., two classes are merged with each other). If the input models are of different types (e.g., an aspect is applied to a class), then asymmetric composition is probably supported by the approach.

Semantics-based composition rules/operators are supported by quite a few modeling approaches (i.e., 9 out of the 14), but this amounts to only a small number of composition rules/operators (23%). In syntax-based composition, the composition is based on syntactic references to the input models. In the context of the composition of crosscutting concerns, this may lead to the well-known fragile pointcut problem, where structural changes in the base concerns may invalidate the compositions. This problem is tackled in semantics-based composition by relying on the meaning of the input models and the relationships to be captured by the composition rather than the structure of the input models or specific naming conventions. Semantics-based composition may be applied to the identification of locations where composition is supposed to occur (e.g., identification of semantically equivalent patterns) or the composition itself (e.g., in simplifying complex results by recognizing redundant model elements; an example is composing inheritance classes that also have a direct relation – simple composition will result in multiple direct relations, all except the first of which are redundant).

Finally, all modeling approaches make use of deterministic composition rules/operators, i.e., the outcome of the composition is fully predictable.

The presented analysis of composition rules/operators is only the first stepping stone for a more in-depth analysis that seeks to discover identical or near-identical composition rules/operators across two or more modeling approaches. Consequently, groups of commonly used and model type-specific composition rules/operators could be established. Such a grouping could inform how difficult it is to combine modeling approaches. Furthermore, support for a specific group of composition rules/operators could play a role when choosing between modeling approaches.

5. CONCLUSION AND FUTURE WORK

The CMA'12 workshop resulted in a significant increase in the number of assessments for modeling approaches, more than tripling the available assessments from 6 to 19 and more than doubling the available models of the bCMS case study from 6 to 14. The modeling approaches have been grouped and related to each other (see Figure 2). Furthermore, the modeling approaches have been contrasted with each other in terms of their software development phases and activities (see Figure 1), paradigms and level of formality (see Figure 3), and their use of composition rules and operators (see Figure 4 and the appendix).

The CMA'12 workshop continued to normalize our understanding of the comparison criteria through in-depth discussions based on the existing assessments, particularly focusing on the issue of composability.

We envision four areas of future work. The first consists of continued surveys of different modeling approaches (e.g., KAOS, SDL, MSC, TTCN, DSLs) in the context of the comparison criteria. An outcome of this area is not only a more uniform platform for comparison, but also the expectation that continued application will provide a testing bed for the criteria themselves, resulting in their continued refinement.

Another area of future work is to continue assessment analysis. The analysis presented in this report is currently at an initial stage. We plan further analysis, including ensuring consistency across the modeling approaches. Another working theory that should be investigated more thoroughly is whether or not there is a relation between composition operators and the execution semantics of a modeling approach and whether or not composition rules are concepts that are defined statically in the metamodel of a modeling approach. In other words, composition operators transform elements of a modeling approach but are themselves not described in the abstract syntax of the modeling approach (e.g., class merge is a composition operator that can be applied to class diagrams but the metamodel for class diagrams does not define the concept of class merge – class merge is defined on top of the metamodel). Composition rules, on the other hand, are part of the language (e.g., the generalization composition rule is defined as a concept in the metamodel for class diagrams, but the corresponding generalization composition operator is not).

The goal of these and other analyses is to identify those criteria that are most useful to the comparison of modeling approaches. We are particularly interested in criteria that will be useful to persons looking for modeling approaches that are most applicable to particular situations, or researchers developing new approaches, to determine existing work. A related area of work is to develop a tool that captures assessment information and provides such searching capabilities.

A third area of continued interest is to use the assessments to postulate where different approaches could be synergistically combined into comprehensive, end-to-end modeling techniques.

Finally, we intend to refine the presentation of the comparison criteria, in terms of their explanations, definitions, and examples. Part of this work includes defining criteria that have not been addressed to date, either through examples or more formal definitions. The items falling into this category that have been identified previously are reusability, scalability, inter-module dependency and interaction, abstraction, usability, ease of evolution, reduction of modeling effort, completeness, and expressiveness.

6. REFERENCES

[1] AOM Bellairs 2011 workshop, http://www.cs.mcgill.ca/~joerg/SEL/AOM_Bellairs_2011.html

[2] AOM Bellairs 2012 workshop, http://www.cs.mcgill.ca/~joerg/SEL/AOM_Bellairs_2012.html

[3] CMA'11 workshop, http://cserg0.site.uottawa.ca/cma2011/

[4] bCMS case study document, http://cserg0.site.uottawa.ca/cma2012/CaseStudy.pdf

[5] Kienzle, J., Guelfi, N., Mustafiz, S.: Crisis Management Systems: A Case Study for Aspect-Oriented Modeling. Katz, S., Mezini, M., Kienzle, J. (eds.) Transactions on Aspect-Oriented Software Development VII, Springer, LNCS vol. 6210:1–22, DOI:10.1007/978-3-642-16086-8_1 (2010)

[6] CMA'12 comparison criteria document, http://cserg0.site.uottawa.ca/cma2012/ComparisonCriteria(post-workshop).pdf

[7] Mussbacher, G., Al Abed, W., Alam, O., Ali, S., Beugnard, A., Bonnet, V., Bræk, R., Capozucca, A., Cheng, B.H.C., Fatima, U., France, R., Georg, G., Guelfi, N., Istoan, P., Jézéquel, J.-M., Kienzle, J., Klein, J., Lézoray, J.-B., Malakuti, S., Moreira, A., Phung-Khac, A., and Troup, L.: Comparing Six Modeling Approaches. 1st Comparing Modeling Approaches Workshop (CMA 2011), Wellington, New Zealand, October 2011. Kienzle, J. (Ed.), Models in Software Engineering, Workshops and Symposia at MODELS 2011, Reports and Revised Selected Papers, Springer, LNCS 7167:217–243, DOI:10.1007/978-3-642-29645-1_22 (2012)

[8] CMA'12 workshop, http://cserg0.site.uottawa.ca/cma2012/

[9] ReMoDD CMA'12 website, http://www.cs.colostate.edu/remodd/v1/content/cmamodels2012

[10] Georg, G.: Activity Theory and its Applications in Software Engineering and Technology. Colorado State University Technical Report CS-11-101 (2011)

[11] Engeström, Y.: Learning by expanding. Helsinki: Orienta-Konsultit (1987)

[12] Unpublished work relating the synergistic application of AT and URN by Georg, G., Mussbacher, G., Troup, L., Amyot, D., France, R., Petriu, D., and Lozano-Fuentes, S. (2012)

[13] Luckey, M., Nagel, B., Gerth, C., and Engels, G.: Adapt Cases: Extending use cases for Adaptive Systems. 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS'11), Waikiki, Honolulu, Hawaii, USA, May 2011. ACM, pp. 30–39, DOI:10.1145/1988008.1988014 (2011)

[14] Mutz, F.: Modeling Structural and Behavioral Adaptation of Software Systems. Master's thesis, University of Paderborn (2012)

[15] Luckey, M. and Engels, G.: High-Quality Specification of Self-Adaptive Software Systems. 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS'13), San Francisco, California, USA, May 2013. ACM (to appear)

[16] ITU-T: Recommendation Z.151 (10/12): User Requirements Notation (URN) - Language Definition. Geneva, Switzerland, http://www.itu.int/rec/T-REC-Z.151/en (2012)

[17] Mussbacher, G.: Aspect-oriented User Requirements Notation. PhD thesis, School of Information Technology and Engineering, University of Ottawa, Canada (2010)

[18] Mussbacher, G., Amyot, D., Araújo, J., and Moreira, A.: Requirements Modeling with the Aspect-oriented User Requirements Notation (AoURN): A Case Study. Katz, S., Mezini, M., and Kienzle, J. (Eds.), Transactions on Aspect-Oriented Software Development (TAOSD) VII, Springer, LNCS 6210:23–68, DOI:10.1007/978-3-642-16086-8_2 (2010)

[19] Mussbacher, G., Araújo, J., Moreira, A., and Amyot, D.: AoURN-based Modeling and Analysis of Software Product Lines. Software Quality Journal, Springer 20(3-4):645–687, DOI:10.1007/s11219-011-9153-8 (2011)

[20] jUCMNav Version 5.2, University of Ottawa, http://jucmnav.softwareengineering.ca/ucm/bin/view/ProjetSEG/WebHome (2012)

[21] Ali, S., Briand, L., and Hemmati, H.: Modeling Robustness Behavior Using Aspect-Oriented Modeling to Support Robustness Testing of Industrial Systems. Journal of Software and Systems Modeling (SoSyM) 11(4):633-670, Springer, DOI:10.1007/s10270-011-0206-z (2012)

[22] Ali, S., Briand, L., Arcuri, A., and Walawege, S.: An Industrial Application of Robustness Testing using Aspect-Oriented Modeling, UML/MARTE, and Search Algorithms. ACM/IEEE 14th International Conference on Model Driven Engineering Languages and Systems (Models 2011), Wellington, New Zealand. Whittle, J., Clark, T., and Kühne, T. (Eds.), Springer, LNCS 6981:108–122, DOI:10.1007/978-3-642-24485-8_9 (2011)

[23] Ali, S. and Yue, T.: Comprehensively Evaluating Conformance Error Rates of Applying Aspect State Machines for Robustness Testing. International Conference on Aspect-Oriented Software Development (AOSD 2012), ACM, pp. 155–166, DOI:10.1145/2162049.2162068 (2012)

[24] Ali, S. and Yue, T.: Studying the Understandability of Aspect State Machines through the Weaving Activity. 19th Asia-Pacific Software Engineering Conference (APSEC 2012). Leung, K. and Muenchaisri, P. (Eds.), IEEE (to appear)

[25] Ali, S., Yue, T., Briand, L., and Walawege, S.: A Product Line Modeling and Configuration Methodology to Support Model-based Testing: An Industrial Case Study. ACM/IEEE 15th International Conference on Model Driven Engineering Languages & Systems (MODELS 2012), Springer, LNCS 7590:726–742, DOI:10.1007/978-3-642-33666-9_46 (2012)

[26] Yue, T. and Ali, S.: Bridging the Gap between Requirements and Aspect State Machines to Support Non-Functional Testing: Industrial Case Studies. 8th European Conference on Modelling Foundations and Applications (ECMFA 2012), Springer, LNCS 7349:133–145, DOI:10.1007/978-3-642-31491-9_12 (2012)

[27] Ali, S., Yue, T., and Briand, L.: Does Aspect-Oriented Modeling Help Improve the Readability of UML State Machines? Journal of Software and Systems Modeling (SoSyM), DOI:10.1007/s10270-012-0293-5 (accepted for publication)

[28] Ali, S., Yue, T., and Briand, L.: Empirically Evaluating the Impact of Applying Aspect State Machines on Modeling Quality and Effort. Simula Research Laboratory, Technical Report (2011-06) (2011)

[29] Yu, E.: Modelling Strategic Relationships for Process Reengineering. PhD Thesis, Graduate Department of Computer Science, University of Toronto, Toronto, Canada (1995)

[30] Chung, L., Nixon, B., Yu, E., and Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers (2000)

[31] Oliveira, A.P.A., Leite, J.C.S.P., Cysneiros, L.M., and Cappelli, C.: Eliciting Multi-Agents Systems Intentionality: From Language Extended Lexicon to i* Models. XXVI International Conference of the Chilean Computer Science Society. IEEE CS Press, pp. 40–49 (2007)

[32] Oliveira, A.P.A. and Cysneiros, L.M.: Defining Strategic Dependency Situations in Requirements Elicitation. IX Workshop on Requirements Engineering (WER), Brazil (2006)

[33] Oliveira, A.P.A., Leite, J.C.S.P., Cysneiros, L.M., and Lucena, C.J.: i* Diagnoses: A Quality Process for Building i* Models. CAiSE'08 Forum, Montpellier, France, June 2008. CEUR Workshop Proceedings 344, pp. 9–12 (2008)

[34] Oliveira, A.P.A.: Intentional Requirements Engineering: A Method for Requirements Elicitation, Modeling, and Analysis. Doctoral Thesis, Computer Science Department, PUC-Rio (2008)

[35] Oliveira, A.P.A., Leite, J.C.S.P., and Cysneiros, L.M.: ERi*c Method - Intentional Requirements Engineering"; XI Workshop on Requirements Engineering (WER), Barcelona, Spain (2008)

[36] Oliveira, A.P.A., Leite, J.C.S.P., and Cysneiros, L.M.: Using i* Meta Modeling for Verifying i* Models. Fourth International i* Workshop (iStar'10), Hammamet, Tunisia (2010)

[37] Yu, E.: Towards modelling and reasoning support for early-phase requirements engineering. Third IEEE International Symposium on Requirements Engineering (ISRE 1997), Annapolis, MD, USA. IEEE CS Press, pp. 226–235, DOI:10.1109/ISRE.1997.566873 (1997)

[38] i* Wiki Guidelines: http://istar.rwth-aachen.de/tiki-index.php?page=i%2A+Guides

[39] Horkoff, J. and Yu, E.: Interactive Analysis of Agent-Goal Models in Enterprise Modeling. International Journal of Information System Modeling and Design, IGI-Global 1(4):1–23, DOI:10.4018/jismd.2010100101 (2010)

[40] Jézéquel, J.-M., Barais, O., and Fleurey, F.: Model Driven Language Engineering with Kermeta. Fernandes, J.M., Lammel, R., Saraiva, J. and Visser, J. (Eds.) 3rd Summer School on Generative and Transformational Techniques in Software Engineering. Springer, LNCS 6491: 201–221, DOI:10.1007/978-3-642-18023-1_5 (2010)

[41] Moha, N., Sen, S., Faucher, C., Barais, O. and Jézéquel, J.-M.: Evaluation of kermeta for solving graph-based problems. International Journal on Software Tools for Technology Transfer (STTT) 12(3–4):273–285, DOI:10.1007/s10009-010-0150-1 (2010)

[42] Muller, P.-A., Fleurey, F., and Jézéquel, J.-M.: Weaving executability into object-oriented meta-languages. Kent, S. and Briand, L. (Eds.) MODELS/UML'2005, Springer, LNCS 3713: 264–278, Montego Bay, Jamaica, DOI:10.1007/11557432_19 (2005)

[43] Kermeta website, http://www.kermeta.org

[44] Barn, B.S. and Clark, T.: Goal Based Alignment of Enterprise Architectures. ICSOFT 2012, pp. 230–236 (2012)

[45] Clark, T. and Barn, B.S.: A common basis for modelling service-oriented and event-driven architecture. ISEC 2012, pp. 23–32, DOI:10.1145/2134254.2134258 (2012)

[46] Clark, T., Barn, B.S., and Oussena, S.: A Method for Enterprise Architecture Alignment. PRET 2012, pp. 48–76, DOI:10.1007/978-3-642-31134-5_3 (2012)

[47] Clark, T., Barn, B.S., and Oussena, S.: LEAP: a precise lightweight framework for enterprise architecture. ISEC 2011, pp. 85–94, DOI:10.1145/1953355.1953366 (2011)

[48] Castejón, H.N.: Collaborations in Service Engineering: Modeling, Analysis and Execution. PhD thesis, Department of Telematics, Norwegian University of Science and Technology (NTNU) (2008)

[49] Kathayat, S.B.: On the Development of Situated Collaborative Services. PhD thesis, Department of Telematics, Norwegian University of Science and Technology (NTNU) (2012)

[50] Object Management Group: Unified Modeling Language (UML) 2.4.1, http://www.omg.org/spec/UML/2.4.1 (2011)

[51] Arctis Developer Reference, http://reference.bitreactive.com/

[52] Woodside, M., Petriu, D.C., Petriu, D.B., Shen, H., Israr, T., and Merseguer, J.: Performance by Unified Model Analysis (PUMA). WOSP'2005, pp. 1–12 (2005)

[53] Alhaj, M. and Petriu, D.C.: Approach for generating performance models from UML models of SOA systems. CASCON'2010, Toronto, Canada, pp. 1–4 (2010)

[54] Alhaj, M. and Petriu, D.C.: Aspect-oriented Modeling of Platforms in Software and Performance Models. 2012 International Conference on Electrical and Computer Systems (ICECS'12) (2012)

[55] Object Management Group, Service Oriented Architecture Modeling Language (SoaML) 1.0.1, http://www.omg.org/spec/SoaML/1.0.1/ (2012)

[56] Object Management Group, UML Profile for MARTE (Modeling and Analysis of Real-Time and Embedded Systems) 1.1, http://www.omg.org/spec/MARTE/1.1/ (2011)

[57] Kienzle, J., Al Abed, W., and Klein, J.: Aspect-Oriented Multi-View Modeling. 8th International Conference on Aspect-Oriented Software Development (AOSD'09), Charlottesville, VA, USA, ACM Press, pp. 89–98, DOI:10.1145/1509239.1509252 (2009)

[58] Kienzle, J., Al Abed, W., Fleurey, F., Jézéquel, J.-M., and Klein, J.: Aspect-Oriented Design with Reusable Aspect Models. Katz, S., Mezini, M., and Kienzle, J. (Eds.), Trans-

actions on Aspect-Oriented Software Development (TAOSD) VII, Springer, LNCS 6210:272–320, DOI:10.1007/978-3-642-16086-8_8 (2010)

[59] Kienzle, J., Duala-Ekoko, E., and Gélineau, S.: AspectOPTIMA: A Case Study on Aspect Dependencies and Interactions. Rashid, A. and Ossher, H. (Eds.), Transactions on Aspect-Oriented Software Development (TAOSD) V, Springer, LNCS 5490:187–234 (2009)

[60] Kramer, M.: Mapping Reusable Aspect Models To Aspect-Oriented Code. Study Thesis, Karlsruhe Institute of Technology (2011)

[61] Al Abed, W., Bonnet, V., Schöttle, M., Yildirim, E., Alam, O., and Kienzle, J.: TouchRAM: A Multitouch-Enabled Tool for Aspect-Oriented Software Design. SLE 2012, Springer, LNCS 7745:275–285, DOI:10.1007/978-3-642-36089-3_16 (2013)

[62] Umple website, http://umple.org/

[63] Forward, A., Badreddin, O., Lethbridge, T.C., and Solano, J.: Model-Driven Rapid Prototyping with Umple. Software Practice and Experience (SPE) 42(7)781–797, DOI:10.1002/spe.1155 (2011)

[64] Lethbridge, T.C., Forward, A. and Badreddin, O.: Umplification: Refactoring to Incrementally Add Abstraction to a Program. 17th Working Conference on Reverse Engineering (WCRE), Boston, MA, USA, October 2010, pp. 220–224, DOI:10.1109/WCRE.2010.32 (2010)

[65] Visual Contract Builder tool website, http://vcl.gforge.uni.lu

[66] Amálio, N., Glodt, C., and Kelsen, P.: Building VCL Models and Automatically Generating Z Specifications from Them. FM 2011. Springer, LNCS 6664:149–153, DOI:10.1007/978-3-642-21437-0_13 (2011)

[67] Amálio, N. and Kelsen, P.: Modular Design by Contract Visually and Formally using VCL. VL/HCC 2010. IEEE, pp. 227–234, DOI:10.1109/VLHCC.2010.39 (2010)

[68] Amálio, N., Kelsen, P., Ma, Q., and Glodt, C.: Using VCL as an Aspect-Oriented Approach to Requirements Modelling. Katz, S., Mezini, M., and Kienzle, J. (Eds.), Transactions on Aspect-Oriented Software Development (TAOSD) VII, Springer, LNCS 6210:151–199, DOI:10.1007/978-3-642-16086-8_5 (2010)

APPENDIX: EXCERPTS FROM ASSESSMENTS RELATED TO COMPOSABILITY

Activity Theory **2. Key Modeling Concepts** **2.2 Composability: composition rules and composition operators**		1) eleInASD metamodel relation	2) refDiagParDoL metam. rel.	3) enableEleReqOut metam. rel.	4) refEleParEle metam. rel.	5) colDiag metamodel relation
B. What is the name of the composition?						
D. Is the composition a:	Composition Rule	x	x	x	x	x
	Composition Operator					
E. If the composition is a composition rule, what other compositions are used to realize the composition defined by the composition rule?						
H. State the signature of the composition…	**eleInASD)** ASD x Element → ASD' **refDiagParDoL)** ASD x DivisionOfLabor → ASD' **enableEleReqOut)** M x ASD_1 x Outcome x ASD_2 x Element → M' **refEleParEle)** ASD x $Element_1$ x $Element_2$ → ASD' **colDiag)** ASD x Collection → Collection'					
I. Does the result of the composition contain a modeling element that does not exist in the source models (i.e., does the composition add new model element(s) to the source models)?	Yes					
	No	x	x	x	x	x
J. Does the composition realize one or more composition rules?	Yes					
	No	x	x	x	x	x
K. What is the mechanism for identifying the inputs for the composition?	Explicit	x	x	x	x	x
	Pattern Matching					
	Binding					
L. What is the mechanism for applying the composition?	Explicit	x	x	x	x	x
	Implicit					
M. Is the composition:	Symmetric					
	Asymmetric	x	x	x	x	x
N. Is the composition:	Syntax-based	x	x	x	x	x
	Semantics-based					
O. Is the composition:	Deterministic	x	x	x	x	x
	Probabilistic					
	Fuzzy					
P. If the composition is a binary composition operator, what algebraic properties does the composition operator provide?	Commutativity					
	Associativity	x	x	x	x	x
	Transitivity		x	x	x	
Q. If the composition specification is a composition operator, does the composition operator produce models that are closed under the operator?	Yes	x	x	x	x	x
	No					
R. Is the intent of the composition to address crosscutting concerns?	Yes					
	No	x	x	x	x	x
S. Is it necessary for a language of the modeling approach to support an explicit ordering of the composition?	Yes					
	No	x	x	x	x	x
T. Is the composition itself separated from the specification of first class entities?	Yes					
	No	x	x	x	x	x
U. How is a composed model intended to be presented to the modeler by a tool? The composed model is intended to be…	… shown with automatic layout.					
	… shown without automatic layout.					
	… shown by annotating the original model.					
	... not shown.					

Adapt Case

2. Key Modeling Concepts

2.2 Composability: composition rules and composition operators

Question	Option	1) Adapt	2) ApplyAdaptation
B. What is the name of the composition?			
D. Is the composition a:	Composition Rule	x	
	Composition Operator		x
E. If the composition is a composition rule, what other compositions are used to realize the composition defined by the composition rule?		(2)	
H. State the signature of the composition…	**Adapt)** Adapt Case Model x Context Model → Context Model' **ApplyAdaptation)** Adapt Case Model x Context Model → Context Model'		
I. Does the result of the composition contain a modeling element that does not exist in the source models (i.e., does the composition add new model element(s) to the source models)?	Yes		
	No	x	x
J. Does the composition realize one or more composition rules?	Yes		x
	No	x	
K. What is the mechanism for identifying the inputs for the composition?	Explicit	x	x
	Pattern Matching		
	Binding		
L. What is the mechanism for applying the composition?	Explicit	x	x
	Implicit		
M. Is the composition:	Symmetric		
	Asymmetric	x	x
N. Is the composition:	Syntax-based		
	Semantics-based	x	x
O. Is the composition:	Deterministic	x	x
	Probabilistic		
	Fuzzy		
P. If the composition is a binary composition operator, what algebraic properties does the composition operator provide?	Commutativity		
	Associativity		
	Transitivity		
Q. If the composition specification is a composition operator, does the composition operator produce models that are closed under the operator?	Yes		x
	No		
R. Is the intent of the composition to address crosscutting concerns?	Yes	x	x
	No		
S. Is it necessary for a language of the modeling approach to support an explicit ordering of the composition?	Yes		
	No	x	x
T. Is the composition itself separated from the specification of first class entities?	Yes		
	No	x	x
U. How is a composed model intended to be presented to the modeler by a tool? The composed model is intended to be…	… shown with automatic layout.		
	… shown without automatic layout.		
	… shown by annotating the original model.		
	... not shown.	x	x

Aspect-oriented User Requirements Notation (AoURN) **2. Key Modeling Concepts** **2.2 Composability: composition rules and composition operators**		1) Grouping	2) Containment	3) Dependency Link	4) GRL Link	5) Hierarchical Decomposition	6) Control Flow Construct	7) AoGRL Composition	8) AoUCM Composition	9) AoGRL Aspect Marker Insertion	10) AoUCM Aspect Marker Insertion
B. What is the name of the composition?											
D. Is the composition a:	Composition Rule							x	x		
	Composition Operator	x	x	x	x	x	x			x	x
E. If the composition is a composition rule, what other compositions are used to realize the composition defined by the composition rule?								(4,9)	(10)		
H. State the signature of the composition…	**Grouping)** 1) Concern$_1$ x Concern$_2$ → Concern'$_1$; 2) Concern x Stakeholder → Concern'; 3) Concern x Aspect Graph → Concern'; 4) Concern x Pointcut Graph → Concern'; 5) Concern x Map → Concern'; 6) Concern x Aspect Map → Concern'; 7) Concern x Pointcut Map → Concern'; **Containment)** 1) Stakeholder x Intentional Element → Stakeholder'; 2) Aspect Graph x Intentional Element → Aspect Graph'; 3) Pointcut Graph x Intentional Element → Pointcut Graph'; 4) Map x Path → Map'; 5) Aspect Map x Path → Aspect Map'; 6) Pointcut Map x Path → Pointcut Map'; 7) Component x Map Element → Component'; **Dependency Link)** 1) M x Stakeholder$_1$ x M x Stakeholder$_2$ → M'; 2) M x Stakeholder x M x Intentional Element → M'; 3) M x Intentional Element x M x Stakeholder → M'; 4) M x Intentional Element$_1$ x M x Intentional Element$_2$ → M'; **GRL Link)** 1) M x Intentional Element$_1$ x M x Intentional Element$_2$ → M'; **Hierarchical Decomposition)** 1) M x Set of Map Elements x M x Set of Start/End Points → M'; **Control Flow Construct)** 1) M x Map Element$_1$ x M x Map Element$_2$ → M'; **AoGRL Composition)** 1) M x Pointcut Graph → M'; **AoUCM Composition)** 1) M x Aspect Map (including its Pointcut Maps) → M'; **AoGRL Aspect Marker Insertion)** 1) M x Intentional Element$_1$ x M x Intentional Element$_2$ → M'; **AoUCM Aspect Marker Insertion)** 1) M x Map Element x Before/After x Start Point/out-path of pointcut stub x End Point/in-path of pointcut stub → M';										
I. Does the result of the composition contain a modeling element that does not exist in the source models (i.e., does the composition add new model element(s) to the source models)?	Yes			x	x	x	x	x	x	x	x
	No	x	x								
J. Does the composition realize one or more composition rules?	Yes				x					x	x
	No	x	x	x		x	x	x	x		
K. What is the mechanism for identifying the inputs for the composition?	Explicit	x	x	x	x	x	x	x	x	x	x
	Pattern Matching							x	x	x	x
	Binding										
L. What is the mechanism for applying the composition?	Explicit	x	x	x	x	x	x	x	x		
	Implicit									x	x
M. Is the composition:	Symmetric										
	Asymmetric	x	x	x	x	x	x	x	x	x	x
N. Is the composition:	Syntax-based	x	x	x	x	x	x	x	x	x	x
	Semantics-based							x	x		
O. Is the composition:	Deterministic	x	x	x	x	x	x	x	x	x	x
	Probabilistic										
	Fuzzy										
P. If the composition is a binary composition operator, what algebraic properties does the composition operator provide?	Commutativity										
	Associativity	x	x	x	x	x	x			x	
	Transitivity	x		x	x	x	x			x	
Q. If the composition specification is a composition operator, does the composition operator produce models that are closed under the operator?	Yes	x	x	x	x	x	x			x	x
	No										
R. Is the intent of the composition to address crosscutting concerns?	Yes	x						x	x	x	x
	No		x	x	x	x	x				
S. Is it necessary for a language of the modeling approach to support an explicit ordering of the composition?	Yes							x	x		
	No	x	x	x	x	x	x			x	x

Aspect-oriented User Requirements Notation (AoURN) (continued)		1) Grouping	2) Containment	3) Dependency Link	4) GRL Link	5) Hierarchical Decomposition	6) Control Flow Construct	7) AoGRL Composition	8) AoUCM Composition	9) AoGRL Aspect Marker Insertion	10) AoUCM Aspect Marker Insertion
T. Is the composition itself separated from the specification of first class entities?	Yes			x					x	x	x
	No	x	x		x	x	x	x	x		
U. How is a composed model intended to be presented to the modeler by a tool? The composed model is intended to be…	… shown with automatic layout.							x	x		x
	… shown without automatic layout.	x	x	x	x	x	x				
	… shown by annotating the original model.							x		x	
	... not shown.										

GRL Link: dependency, contribution, correlation, decomposition
Hierarchical Decomposition: static/dynamic/sychronizing/blocking stubs
Control Flow Construct: sequence, OR/AND-fork, OR/AND-join, waiting place/timer, failure point, abort/failure start points

AspectSM

2. Key Modeling Concepts

2.2 Composability: composition rules and composition operators

Question	Option	1) Pointcut together with Advice/Introduction, Weaving-directive state machine	2) Weaving-directive state machine
B. What is the name of the composition?			
D. Is the composition a:	Composition Rule	x	(*)
	Composition Operator		
E. If the composition is a composition rule, what other compositions are used to realize the composition defined by the composition rule?		Advice and Introduction	Aspect state machines
H. State the signature of the composition…	**Pointcut...)** UMLStateMachine x Aspect State Machine → Woven State Machine **Weaving-directive...)** UMLStateMachine x Aspect State Machine x Weaving Directive State machine → Woven State Machine		
I. Does the result of the composition contain a modeling element that does not exist in the source models (i.e., does the composition add new model element(s) to the source models)?	Yes		
	No	x	x
J. Does the composition realize one or more composition rules?	Yes		
	No	x	x
K. What is the mechanism for identifying the inputs for the composition?	Explicit	x	x
	Pattern Matching		
	Binding		
L. What is the mechanism for applying the composition?	Explicit	x	x
	Implicit		
M. Is the composition:	Symmetric		
	Asymmetric	x	x
N. Is the composition:	Syntax-based	x	x
	Semantics-based		
O. Is the composition:	Deterministic	x	x
	Probabilistic		
	Fuzzy		
P. If the composition is a binary composition operator, what algebraic properties does the composition operator provide?	Commutativity		
	Associativity		
	Transitivity		
Q. If the composition specification is a composition operator, does the composition operator produce models that are closed under the operator?	Yes		
	No		x
R. Is the intent of the composition to address crosscutting concerns?	Yes	x	
	No		x
S. Is it necessary for a language of the modeling approach to support an explicit ordering of the composition?	Yes	x	x
	No		
T. Is the composition itself separated from the specification of first class entities?	Yes		
	No	x	x
U. How is a composed model intended to be presented to the modeler by a tool? The composed model is intended to be…	… shown with automatic layout.	x	x
	… shown without automatic layout.		
	… shown by annotating the original model.		
	... not shown.		

(*) neither: control structure of UML state machines

Intentional Requirements Engineering

2. Key Modeling Concepts

2.2 Composability: composition rules and composition operators

Question	Option	1) SDsituation	2) SR construct
B. What is the name of the composition?			
D. Is the composition a:	Composition Rule	x	x
	Composition Operator		
E. If the composition is a composition rule, what other compositions are used to realize the composition defined by the composition rule?		(*)	(**)
H. State the signature of the composition…	**SDsituation)** $SDsituation_1$ x $SDsituation_2$ x .... $SDsituation_Z$ = Organizational Problem (SDsituation Diagram) **SR construct)** $TaskMean_1$ (task $decomposition_1$) or .... $TaskMean_Z$ (task $decomposition_Z$) = SR construct		
I. Does the result of the composition contain a modeling element that does not exist in the source models (i.e., does the composition add new model element(s) to the source models)?	Yes	x	x
	No		
J. Does the composition realize one or more composition rules?	Yes	x	
	No		x
K. What is the mechanism for identifying the inputs for the composition?	Explicit	x	x
	Pattern Matching		
	Binding		
L. What is the mechanism for applying the composition?	Explicit	x	x
	Implicit		
M. Is the composition:	Symmetric		
	Asymmetric	x	x
N. Is the composition:	Syntax-based		
	Semantics-based	x	x
O. Is the composition:	Deterministic		
	Probabilistic		
	Fuzzy		
P. If the composition is a binary composition operator, what algebraic properties does the composition operator provide?	Commutativity		
	Associativity	x	x
	Transitivity		
Q. If the composition specification is a composition operator, does the composition operator produce models that are closed under the operator?	Yes		
	No		
R. Is the intent of the composition to address crosscutting concerns?	Yes		
	No		
S. Is it necessary for a language of the modeling approach to support an explicit ordering of the composition?	Yes	x	x
	No		
T. Is the composition itself separated from the specification of first class entities?	Yes		
	No	x	x
U. How is a composed model intended to be presented to the modeler by a tool? The composed model is intended to be…	… shown with automatic layout.		
	… shown without automatic layout.	x	x
	… shown by annotating the original model.		
	... not shown.		

(*) SR constructs are inside of SDsituations.
(**) i* Framework links (means-end, contribution, and task decomposition)

i*

2. Key Modeling Concepts

2.2 Composability: composition rules and composition operators

Question	Option	1) Actor boundary	2) i* link/relationship (contribution, decomposition, means-ends, dependency)
B. What is the name of the composition?			
D. Is the composition a:	Composition Rule	x	x
	Composition Operator		
E. If the composition is a composition rule, what other compositions are used to realize the composition defined by the composition rule?			
H. State the signature of the composition…	**Actor boundary)** M x Actor x Element$_1$ x … x Element$_n$ -> M' **i* link/relationship)** M x Element$_1$ x … x Element$_n$ -> M'		
I. Does the result of the composition contain a modeling element that does not exist in the source models (i.e., does the composition add new model element(s) to the source models)?	Yes		x
	No	x	
J. Does the composition realize one or more composition rules?	Yes		
	No	x	x
K. What is the mechanism for identifying the inputs for the composition?	Explicit	x	x
	Pattern Matching		
	Binding		
L. What is the mechanism for applying the composition?	Explicit	x	x
	Implicit		
M. Is the composition:	Symmetric		
	Asymmetric	x?	x?
N. Is the composition:	Syntax-based	x	x
	Semantics-based	x	x
O. Is the composition:	Deterministic	x	x
	Probabilistic		
	Fuzzy		
P. If the composition is a binary composition operator, what algebraic properties does the composition operator provide?	Commutativity		
	Associativity		
	Transitivity		x
Q. If the composition specification is a composition operator, does the composition operator produce models that are closed under the operator?	Yes		
	No	x	x
R. Is the intent of the composition to address crosscutting concerns?	Yes		
	No	x	x
S. Is it necessary for a language of the modeling approach to support an explicit ordering of the composition?	Yes		
	No	x	
T. Is the composition itself separated from the specification of first class entities?	Yes		x
	No	x	
U. How is a composed model intended to be presented to the modeler by a tool? The composed model is intended to be…	… shown with automatic layout.		
	… shown without automatic layout.	x	x
	… shown by annotating the original model.	x	x
	... not shown.		

Kermeta

2. Key Modeling Concepts

2.2 Composability: composition rules and composition operators

Question	Option	1) require	2) uses	3) aspect
B. What is the name of the composition?				
D. Is the composition a:	Composition Rule			
	Composition Operator	x	x	x
E. If the composition is a composition rule, what other compositions are used to realize the composition defined by the composition rule?				
H. State the signature of the composition…	**require)** $ModelingUnit_1$ x … x $ModelingUnit_n$ → $ModelingUnit'_1$ **uses)** $Package_1$ x … x $Package_n$ → $Package'_1$ **aspect)** $Class_1$ x $Class_2$ -> $Class'_1$			
I. Does the result of the composition contain a modeling element that does not exist in the source models (i.e., does the composition add new model element(s) to the source models)?	Yes			
	No	x	x	x
J. Does the composition realize one or more composition rules?	Yes			
	No	x	x	x
K. What is the mechanism for identifying the inputs for the composition?	Explicit	x	x	
	Pattern Matching			x
	Binding			
L. What is the mechanism for applying the composition?	Explicit	x	x	x
	Implicit			
M. Is the composition:	Symmetric	x		x
	Asymmetric		x	
N. Is the composition:	Syntax-based	x	x	x
	Semantics-based			
O. Is the composition:	Deterministic	x	x	x
	Probabilistic			
	Fuzzy			
P. If the composition is a binary composition operator, what algebraic properties does the composition operator provide?	Commutativity	x		x
	Associativity	x		x
	Transitivity			
Q. If the composition specification is a composition operator, does the composition operator produce models that are closed under the operator?	Yes			x
	No	x	x	
R. Is the intent of the composition to address crosscutting concerns?	Yes			
	No	x	x	x
S. Is it necessary for a language of the modeling approach to support an explicit ordering of the composition?	Yes			
	No	x	x	x
T. Is the composition itself separated from the specification of first class entities?	Yes	x		
	No		x	x
U. How is a composed model intended to be presented to the modeler by a tool? The composed model is intended to be…	… shown with automatic layout.	x		
	… shown without automatic layout.			x
	… shown by annotating the original model.			
	... not shown.		x	

LEAP

2. Key Modeling Concepts

2.2 Composability: composition rules and composition operators

Question	Option	1) Inheritance	2) Reference	3) Connector	4) List Construction	5) Function Application
B. What is the name of the composition?						
D. Is the composition a:	Composition Rule					
	Composition Operator	x	x	x	x	x
E. If the composition is a composition rule, what other compositions are used to realize the composition defined by the composition rule?						
H. State the signature of the composition…	**Inheritance)** Class x Class → Class **Reference)** M x Class x Class → M' **Connector)** M x Component_1 x Output Port x Component_2 X Input Port → M' **List Construction)** Value_1 x … x Value_n → [Value] **Function Application)** Function x Value_1 x ... x Value_n → Value'					
I. Does the result of the composition contain a modeling element that does not exist in the source models (i.e., does the composition add new model element(s) to the source models)?	Yes					
	No	x	x	x	x	x
J. Does the composition realize one or more composition rules?	Yes					
	No	x	x	x	x	x
K. What is the mechanism for identifying the inputs for the composition?	Explicit	x	x	x	x	x
	Pattern Matching					
	Binding					
L. What is the mechanism for applying the composition?	Explicit	x	x	x	x	x
	Implicit					
M. Is the composition:	Symmetric					
	Asymmetric	x	x	x	x	x
N. Is the composition:	Syntax-based	x	x	x		
	Semantics-based				x	x
O. Is the composition:	Deterministic	x	x	x	x	x
	Probabilistic					
	Fuzzy					
P. If the composition is a binary composition operator, what algebraic properties does the composition operator provide?	Commutativity					
	Associativity	x	x	x	x	x
	Transitivity	x				
Q. If the composition specification is a composition operator, does the composition operator produce models that are closed under the operator?	Yes	x	x	x	x	x
	No					
R. Is the intent of the composition to address crosscutting concerns?	Yes					
	No	x	x	x	x	x
S. Is it necessary for a language of the modeling approach to support an explicit ordering of the composition?	Yes	x	x	x	x	x
	No					
T. Is the composition itself separated from the specification of first class entities?	Yes	x	x	x	x	x
	No					
U. How is a composed model intended to be presented to the modeler by a tool? The composed model is intended to be…	… shown with automatic layout.					
	… shown without automatic layout.	x	x	x		
	… shown by annotating the original model.					
	... not shown.					

Model Driven Service Engineering (MDSE)

2. Key Modeling Concepts

2.2 Composability: composition rules and composition operators

Question	Option	1) Collaboration	2) Role binding	3) Activity	4) Class
B. What is the name of the composition?					
D. Is the composition a:	Composition Rule	(*)		(**)	(***)
	Composition Operator		x		
E. If the composition is a composition rule, what other compositions are used to realize the composition defined by the composition rule?					
H. State the signature of the composition…	**Collaboration)** Roles x Collaboration Uses x Links x Activities → Collaboration **Role binding)** Roles x Role → Role **Activity)** Actions x Operations x Flows x Control Nodes x Pseudonodes → Activity **Class)** Parts x Collaboration Uses x Links x Activitie x Attributes x Operations → Class				
I. Does the result of the composition contain a modeling element that does not exist in the source models (i.e., does the composition add new model element(s) to the source models)?	Yes	x		x	x
	No		x		
J. Does the composition realize one or more composition rules?	Yes	x	x	x	x
	No				
K. What is the mechanism for identifying the inputs for the composition?	Explicit	x	x	x	x
	Pattern Matching				
	Binding	x	x		x
L. What is the mechanism for applying the composition?	Explicit	x	x	x	x
	Implicit				
M. Is the composition:	Symmetric	x	x	x	x
	Asymmetric	x		x	x
N. Is the composition:	Syntax-based	x	x	x	x
	Semantics-based				
O. Is the composition:	Deterministic	x	x	x	x
	Probabilistic				
	Fuzzy				
P. If the composition is a binary composition operator, what algebraic properties does the composition operator provide?	Commutativity				
	Associativity				
	Transitivity		x	x	
Q. If the composition specification is a composition operator, does the composition operator produce models that are closed under the operator?	Yes		x		
	No				
R. Is the intent of the composition to address crosscutting concerns?	Yes	x	x		
	No			x	x
S. Is it necessary for a language of the modeling approach to support an explicit ordering of the composition?	Yes	x	x	x	x
	No				
T. Is the composition itself separated from the specification of first class entities?	Yes		x		
	No	x		x	x
U. How is a composed model intended to be presented to the modeler by a tool? The composed model is intended to be…	… shown with automatic layout.				
	… shown without automatic layout.	x		x	x
	… shown by annotating the original model.		x		
	... not shown.		x		

(*) neither: A collaboration is composed from collaboration uses, roles and an activity.
(**) neither: An activity is composed from actions and flows.
(***) neither: A class is composed from parts; links; attributes; operations; and behavior/an activity.

Performance from Unified Modeling Analysis for SOA (PUMA4SOA) **2. Key Modeling Concepts** **2.2 Composability: composition rules and composition operators**		1) Stereotyping	2) Aspect weaving	3) Model Transforma-tion
B. What is the name of the composition?				
D. Is the composition a:	Composition Rule			
	Composition Operator	x	x	x
E. If the composition is a composition rule, what other compositions are used to realize the composition defined by the composition rule?				
H. State the signature of the composition…	**Stereotyping)** UML Model Element x Metadata of MARTE Profile → Annotated UML model **Aspect weaving)** Platform Independent Model x Context Specific Aspect Mode) → Platform Specific Model **Model Transformation)** 1) UML → CSM; 2) CSM → LQN			
I. Does the result of the composition contain a modeling element that does not exist in the source models (i.e., does the composition add new model element(s) to the source models)?	Yes	x		x
	No		x	
J. Does the composition realize one or more composition rules?	Yes		x	
	No	x		x
K. What is the mechanism for identifying the inputs for the composition?	Explicit	x		
	Pattern Matching			
	Binding		x	
L. What is the mechanism for applying the composition?	Explicit	x	x	x
	Implicit			
M. Is the composition:	Symmetric	x	x	
	Asymmetric			x
N. Is the composition:	Syntax-based		x	
	Semantics-based	x	x	x
O. Is the composition:	Deterministic	x	x	x
	Probabilistic			
	Fuzzy			
P. If the composition is a binary composition operator, what algebraic properties does the composition operator provide?	Commutativity			
	Associativity	x	x	x
	Transitivity			
Q. If the composition specification is a composition operator, does the composition operator produce models that are closed under the operator?	Yes	x	x	
	No			x
R. Is the intent of the composition to address crosscutting concerns?	Yes		x	
	No	x		x
S. Is it necessary for a language of the modeling approach to support an explicit ordering of the composition?	Yes			
	No	x	x	x
T. Is the composition itself separated from the specification of first class entities?	Yes			x
	No		x	
U. How is a composed model intended to be presented to the modeler by a tool? The composed model is intended to be…	… shown with automatic layout.		x	x
	… shown without automatic layout.			
	… shown by annotating the original model.	x		
	... not shown.			

<u>Reusable Aspect Models (RAM)</u> **2. Key Modeling Concepts** **2.2 Composability: composition rules and composition operators**		1) Instantiation (Customization)	2) Instantiation (Extension)	3) Class Merge	4) Message View Inlining	5) Message View Advising	6) State View Advising
B. What is the name of the composition?							
D. Is the composition a:	Composition Rule	x	x		x	x	x
	Composition Operator			x			
E. If the composition is a composition rule, what other compositions are used to realize the composition defined by the composition rule?		(3)	(3)		(*)	(*)	(**)
H. State the signature of the composition…	**Instantiation (Customization))** 1) Aspect$_1$ x Model Element$_1$ x Aspect$_2$ x Model Element$_2$ → Aspect'$_1$ **Instantiation (Extension))** 1) Aspect$_1$ x Aspect$_2$ → Aspect'$_1$ **Class Merge)** 1) Class$_1$ x Class$_2$ → Class'$_1$ **Message View Inlining)** 1) Message View$_1$ x Message View$_2$ → Message View'$_1$ **Message View Advising)** 1) Message View$_1$ x Message View$_2$ → Message View'$_1$ **State View Advising)** 1) State View$_1$ x State View$_2$ → State View'$_1$						
I. Does the result of the composition contain a modeling element that does not exist in the source models (i.e., does the composition add new model element(s) to the source models)?	Yes						
	No	x	x	x	x	x	x
J. Does the composition realize one or more composition rules?	Yes			x			
	No	x	x		x	x	x
K. What is the mechanism for identifying the inputs for the composition?	Explicit	x	x	x	x	x	x
	Pattern Matching	x	x			x	x
	Binding	x	x	x			
L. What is the mechanism for applying the composition?	Explicit	x	x		x	x	x
	Implicit			x			
M. Is the composition:	Symmetric		x	x			
	Asymmetric	x			x	x	x
N. Is the composition:	Syntax-based	x	x	x	x	x	x
	Semantics-based					x	
O. Is the composition:	Deterministic	x	x	x	x	x	x
	Probabilistic						
	Fuzzy						
P. If the composition is a binary composition operator, what algebraic properties does the composition operator provide?	Commutativity		x	x			
	Associativity	x	x	x	x		
	Transitivity	x	x	x	x	x	
Q. If the composition specification is a composition operator, does the composition operator produce models that are closed under the operator?	Yes	x	x	x	x	x	x
	No						
R. Is the intent of the composition to address crosscutting concerns?	Yes	x	x	x	x	x	x
	No						
S. Is it necessary for a language of the modeling approach to support an explicit ordering of the composition?	Yes						
	No	x	x	x	x	x	x
T. Is the composition itself separated from the specification of first class entities?	Yes	x	x	x			
	No				x	x	x
U. How is a composed model intended to be presented to the modeler by a tool? The composed model is intended to be…	… shown with automatic layout.	x	x	x	x	x	x
	… shown without automatic layout.						
	… shown by annotating the original model.						
	... not shown.						

(*) Message Merge (not assessed at the moment)
(**) State Merge (not assessed at the moment)

Unified Modeling Language (UML) **(note that this table summarizes six assessments)** **2. Key Modeling Concepts** **2.2 Composability: composition rules and composition operators**		1) AD: control flows (includes data flow and control nodes)	2) CD: Generalization, Association, Aggregation, Composition	3) CoD: connectors	4) SD: Referencing	5) SD: Messaging	6) SD: Follows	7) SM: Submachines/Composite States, Orthogonal States	8) UCD: includes and extends relation-ships
B. What is the name of the composition?									
D. Is the composition a:	Composition Rule	x	x		x	x	x	(**)	(***)
	Composition Operator		x	x					
E. If the composition is a composition rule, what other compositions are used to realize the composition defined by the composition rule?			(*)						
H. State the signature of the composition…	**AD)** Activity x Activity → Activity Diagram **CD: Generalization)** rule: 1) M x $Class_1$ x M x $Class_2$ → M'; operator: 1) M x $Class_1$ x $Class_2$ → M'; **CD: Association, Aggregation, Composition)** rule: 1) M x Class x M x Class → M'; operator: 1) M x Class x Class → M'; **CoD)** Component x Connector x Component → Component Structural Diagram **SD: Referencing)** Reference Fragment x Complete Sequence Diagram **SD: Messaging)** Source Lifeline x Message x Destination Lifeline **SD: Follows)** 1) Fragment x Fragment; 2) Message x Fragment; 3) Fragment x Message **SM)** for submachine/composite state: State Machine x Set (Transitions) → State Machine; for orthogonal states: State Machine x Set (State Machine) → State Machine **UCD)** 1) Use Case x Use Case → Use Case Diagram; 2) Use Case x Actor → Use Case Diagram								
I. Does the result of the composition contain a modeling element that does not exist in the source models (i.e., does the composition add new model element(s) to the source models)?	Yes		x						
	No	x		x	x	x	x	x	x
J. Does the composition realize one or more composition rules?	Yes			x					
	No	x	x		x	x		x	x
K. What is the mechanism for identifying the inputs for the composition?	Explicit	x	x	x	x	x	x	x	x
	Pattern Matching								
	Binding								
L. What is the mechanism for applying the composition?	Explicit	x	x	x	x	x	x	x	x
	Implicit								
M. Is the composition:	Symmetric	x		x					x
	Asymmetric		x		x	x	x	x	
N. Is the composition:	Syntax-based		x		x	x	x	x	x
	Semantics-based	x		x					
O. Is the composition:	Deterministic	x	x	x	x	x	x	x	x
	Probabilistic								
	Fuzzy								
P. If the composition is a binary composition operator, what algebraic properties does the composition operator provide?	Commutativity	x		x					
	Associativity		x						
	Transitivity		x				x		x
Q. If the composition specification is a composition operator, does the composition operator produce models that are closed under the operator?	Yes	x	x	x				x	
	No								
R. Is the intent of the composition to address crosscutting concerns?	Yes				x			x	
	No	x	x	x	x	x	x		x
S. Is it necessary for a language of the modeling approach to support an explicit ordering of the composition?	Yes							x	x
	No	x	x	x	x	x	x		
T. Is the composition itself separated from the specification of first class entities?	Yes			x		x	x		
	No	x	x		x			x	x
U. How is a composed model intended to be presented to the modeler by a tool? The composed model is intended to be…	… shown with automatic layout.					x	x		
	… shown without automatic layout.		x						
	… shown by annotating the original model.								
	... not shown.				x			x	

AD = Activity Diagram, CD = Class Diagram, CoD = Component Diagram, SD = Sequence Diagram, SM = State Machine, UCD = Use Case Diagram
(*) a dedicated composition operator exists for this composition rule
(**) neither: Hierarchical Decomposition
(***) neither: Relationship

Umple

2. Key Modeling Concepts

2.2 Composability: composition rules and composition operators

Question	Option	1) Use	2) Generalization (isA statement)	3) Association	4) Mixin	5) CodeInjection (before and after statements)	6) Variation Point Invocation
B. What is the name of the composition?							
D. Is the composition a:	Composition Rule		x	x			
	Composition Operator	x	x	x	x	x	x
E. If the composition is a composition rule, what other compositions are used to realize the composition defined by the composition rule?			(*)	(**)			
H. State the signature of the composition…	**Use)** File x File → PartialModel **Generalization)** Class x Class → Partial Model **Association)** Class x Class → Partial Model **Mixin)** 1) Class x Class → Class; 2) StateMachine x StateMachine → StateMachine **CodeInjection)** First Class Entity x Code → First Class Entity **Variation Point Invocation)** Concern x VariationPoint → PartialModel						
I. Does the result of the composition contain a modeling element that does not exist in the source models (i.e., does the composition add new model element(s) to the source models)?	Yes	x	x			x	x
	No			x	x		
J. Does the composition realize one or more composition rules?	Yes						
	No			x		x	
K. What is the mechanism for identifying the inputs for the composition?	Explicit	x	x	x	x	x	x
	Pattern Matching	x					
	Binding						
L. What is the mechanism for applying the composition?	Explicit	x	x	x		x	x
	Implicit				x		
M. Is the composition:	Symmetric						
	Asymmetric	x	x	x	x	x	x
N. Is the composition:	Syntax-based	x	x	x	x		x
	Semantics-based					x	
O. Is the composition:	Deterministic	x	x	x	x	x	x
	Probabilistic						
	Fuzzy						
P. If the composition is a binary composition operator, what algebraic properties does the composition operator provide?	Commutativity	x		x			
	Associativity	x	x		x		
	Transitivity	x	x				
Q. If the composition specification is a composition operator, does the composition operator produce models that are closed under the operator?	Yes	x	x		x		
	No			x		x	x
R. Is the intent of the composition to address crosscutting concerns?	Yes				x		
	No	x	x	x		x	x
S. Is it necessary for a language of the modeling approach to support an explicit ordering of the composition?	Yes				x	x	
	No	x	x	x		x	x
T. Is the composition itself separated from the specification of first class entities?	Yes	x					x
	No		x	x	x	x	
U. How is a composed model intended to be presented to the modeler by a tool? The composed model is intended to be…	… shown with automatic layout.		x				
	… shown without automatic layout.	x		x	x		x
	… shown by annotating the original model.						
	... not shown.					x	

(*) When the model is compiled, then the rule takes effect, resulting in many effects, most notably inheritance. (**) When the model is compiled, then the rule takes effect, resulting in many effects such as generation of code implementing networks of constraints.

Visual Contract Language (VCL)

2. Key Modeling Concepts

2.2 Composability: composition rules and composition operators

B. What is the name of the composition?		1) Package incorporation	2) Package merge	3) Package overrides	4) Package extends	5) Class inheritance	6) Assertion and contract importing	7) Integral Extension	8) Merge Extension	9) Concurrent Join Ext.	10) Sequential Join Ext.
D. Is the composition a:	Composition Rule		x	x	x						
	Composition Operator	x				x	x	x	x	x	x
E. If the composition is a composition rule, what other compositions are used to realize the composition defined by the composition rule?			(*)	(*)	(*)						
H. State the signature of the composition…	**Package incorporation)** P IncorporatedPkg x P Merge x P Override x P Extends → NewPkg **Package merge)** Merge == IncorporatedPkgA x IncorporatedPkgB x Set IDs **Package overrides)** Override == IncorporatedPkgA x Set IDs x IncorporatedPkgB **Package extends)** Extends == Set IDs **Class inheritance)** ClassParent x ClassChildOnly → NewClassChild **Assertion and contract importing)** P AssertionOrContract x NewAssertionOrContractOnly → NewAssertionOrContract **Integral Extension)** Pkg x P Operations → P NewOperations **Merge Extension)** P (Pkg x P Operations) → P NewOperations **Concurrent Join Extension)** P (Pkg x P Operations) x JoinContract → P NewOperations **Sequential Join Extension)** P (Pkg x P Operations) x optional BeforeJoinContract x optional AfterJoinContract → P NewOperations										
I. Does the result of the composition contain a modeling element that does not exist in the source models (i.e., does the composition add new model element(s) to the source models)?	Yes										
	No	x	x	x	x	x	x	x	x	x	x
J. Does the composition realize one or more composition rules?	Yes	x									
	No										
K. What is the mechanism for identifying the inputs for the composition?	Explicit	x	x	x	x	x	x	x	x	x	x
	Pattern Matching										
	Binding										
L. What is the mechanism for applying the composition?	Explicit	x	x	x	x	x	x	x	x	x	x
	Implicit										
M. Is the composition:	Symmetric	x	x	x	x	x	x	x	x	x	x
	Asymmetric										
N. Is the composition:	Syntax-based	x	x	x	x	x	x	x	x	x	x
	Semantics-based										
O. Is the composition:	Deterministic	x	x	x	x	x	x	x	x	x	x
	Probabilistic										
	Fuzzy										
P. If the composition is a binary composition operator, what algebraic properties does the composition operator provide?	Commutativity	x									
	Associativity	x									
	Transitivity	x									
Q. If the composition specification is a composition operator, does the composition operator produce models that are closed under the operator?	Yes										
	No										
R. Is the intent of the composition to address crosscutting concerns?	Yes									x	x
	No	x	x	x	x	x	x	x	x		
S. Is it necessary for a language of the modeling approach to support an explicit ordering of the composition?	Yes										x
	No	x	x	x	x	x	x	x	x	x	x
T. Is the composition itself separated from the specification of first class entities?	Yes										
	No	x	x	x	x	x	x	x	x	x	x
U. How is a composed model intended to be presented to the modeler by a tool? The composed model is intended to be…	… shown with automatic layout.										
	… shown without automatic layout.										
	… shown by annotating the original model.										
	... not shown.	x	x	x	x	x	x	x	x	x	x

(*) This is part of package incorporation. When packages are incorporated, merge/override/extends rules are taken into consideration in the incorporation.