

Multi-view Consistency in UML

Alexander Knapp¹ and Till Mossakowski²

¹ Universität Augsburg, Germany

² Otto-von-Guericke Universität Magdeburg, Germany

Abstract. We study the question of consistency of multi-view models in UML and OCL. We first critically survey the large amount of literature that already exists. We find that only limited subsets of the UML/OCL have been covered so far and that consistency checks mostly only cover structural aspects, whereas only few methods also address behaviour. We also give a classification of different techniques for multi-view UML/OCL consistency: consistency rules, the system model approach, dynamic meta-modelling, universal logic, and heterogeneous transformation. Finally, we elaborate cornerstones of a comprehensive distributed semantics approach to consistency using OMG’s Distributed Ontology, Model and Specification Language (DOL).

1 Introduction

Hartmut Ehrig was a researcher whose broad scope of interests ranged from category and automata theory through algebraic specifications and graph grammars to models of concurrency, and in all these fields he achieved fundamental results and contributed far-reaching and novel ideas. It is sad that such a great researcher passed away far too early after his retirement.

One of the many themes of Hartmut Ehrig’s research has been the multi-viewpoint integration in the specification of complex systems [12]. We here address this problem more specifically in the context of the “Unified Modeling Language” (UML [34]). UML is a complex visual language featuring 14 different diagram types which may be complemented by textual annotations in the “Object Constraint Language” (OCL [33]); both languages are standardised by the Object Management Group (OMG). Already for UML 1.1 van Emde Boas observed “that UML is not a single language but a hybrid of several languages” [11] and Cook et al. [7] coined the notion of UML as “a family of languages”. The multitude of diagram types and sub-languages offered by the UML/OCL allows the modeller to reduce the complexity of a model by specifying a system from different viewpoints: data, behaviour, interaction, component architecture, etc. Multi-view modelling and the necessity to integrate views devised from different viewpoints has been intensively discussed in the literature in general by Hartmut Ehrig et al. [12] and others [4], in the software architecture community [8, 21, 22], in the UML community [5], in the SysML community [32, 39], and also in other communities [2, 17, 35].

A central question in multi-view modelling is whether such a family of UML/OCL diagrams and annotations is still consistent, i.e., conjointly instantiable such that all views from all viewpoints are satisfied w.r.t. their (well-defined) semantics [c46]³. This

³ References prefixed with a ‘c’ refer to the multi-view UML/OCL consistency bibliography assembled in a separate list.

consistency problem has already been stated in the early UML days [5, 13, 14], and has been addressed quite intensively in the literature. In particular, several categorisations for partitioning the consistency problem have been designed: Engels et al. [c26] suggest to distinguish between horizontal (or intra-model) and vertical (or inter-model) consistency, i.e., whether the views are on the same level of abstraction; as well as syntactic (structural well-formedness of the abstract syntax) and semantic consistency (compatibility of behaviour). Mens et al. [29] focus more on the intention of sub-languages and give a classification into structural vs. behavioral diagrams and their use on the specification vs. instance level. Allaki et al. [1] combine these schemes into a typological frame of mono- vs. multi-diagram, specification vs. instance, and syntactic vs. semantic, and furthermore add a taxonomy of consistency problems in a terminological dimension, mentioning incompleteness, ambiguity, contradiction, incompatibility, and anomaly. From a verification perspective, Hilken et al. [18] present a list of structural and behavioural verification tasks for UML models considering besides consistency the categories of consequence, independence, executability, and reachability. A structural verification task considers a single (integrated) system state only, whereas a behavioural task pertains to a sequence of states. In contrast to [c46], here “[c]onsistency problems are structural problems and do not involve behaviour” [18, p. 122].

The large number of approaches to multi-view consistency in the literature has also been reviewed and summarised [3, 20, 28, 40, 41, 42]. In particular, Torre et al. [40, 41] systematically survey existing consistency rules. They find that most rules are syntactic (88.21% in [40] and 81.89% according to the more comprehensive [41]), and that most of the rules are related to class diagrams (71.58%), sequence diagrams (47.37%), and state machine diagrams (42.11%). Moreover, they deplore that “it appears that researchers tend to discuss very similar consistency rules, over and over again”, and conclude that “much more work is needed to develop consistency rules for all 14 UML diagrams, in all dimensions of consistency (e.g., semantic and syntactic on the one hand, horizontal, vertical and evolution on the other hand)” [40].

In this paper, we want to take up the challenge posed by Torre et al.: What is needed to develop a notion of multi-view consistency for a multitude of the (and ideally all) 14 UML diagrams and the OCL? We first give a comprehensive overview over the existing approaches to multi-view UML/OCL consistency in the literature both for UML 1 and UML 2 starting from the surveys mentioned above. We list which diagram types and sub-languages of UML/OCL are covered by each approach, which consistency technique it applies, whether it tackles structural or behavioural consistency, and which formalism and tool it uses. Our main contribution here is to point out and survey the variety of techniques to a grip on consistency for a heterogeneous, multi-view language like UML/OCL, ranging from syntactic consistency rules over an overarching, semantic system model to heterogeneous transformations. We find that structural consistency is considered more often by far and that either structural, syntactic consistency rules or an encoding into a system model or some universal logic prevails.

Purely syntactic approaches, however, do not help in ensuring behavioural consistency, and having to encode all UML/OCL sub-languages into a single semantic model or formalism, somewhat neglects the inherent heterogeneity of this multi-view language and almost necessarily leads to a certain unnaturalness. More often than not, it would be

preferable to represent the structural and behavioural semantics of a sub-language in an appropriate semantic domain of its own and only to relate these sub-languages and their semantic domains by translations. We therefore outline, extending our previous work on a truly heterogeneous approach to UML/OCL semantics [c11, c42], a consistency approach based on distributed heterogeneity in the “Distributed Ontology, Model and Specification Language” (DOL), also standardised by the OMG.

2 Review of Approaches to Multi-view Consistency in UML

Tables 1 and 2 contain an overview of existing approaches to multi-view consistency both for UML/OCL 1 (up to 2004) and UML/OCL 2 (starting in 2005), roughly ordered by their date of appearance. While the literature on UML abounds, for our topic, our literature review is sound and also comprehensive. Our starting points were the surveys [3,

Reference	CD	OD	SM	ID	AD	OCL	cons.	class.	Form./Tool
* Egyed [c18, c19]	●	●	●	●			T	s	VIEWINTEGRA
* Große-Rhode [c32, c33]	●		●	●			S	b	transf. syst.
Reggio et al. [c66]	●		●				U	(s/b)	CASL-LTL
McUumber, Cheng [c56]	●		●				U	b	SPIN
* krtUML [c14]	●	●	●				S	s/b	symp. trans. syst.
Bernardi et al. [c7]				●	●		U	(b)	Petri nets
* xUML [c57]	●		●	●	●	●	S	(s)	Exec. UML
* Küster et al. [c23, c26]	●	●	●	●			U	b	CSP/FDR
* Hausmann et al. [c25]			●	●			D	b	Graph transf.
Spanoudakis, Kim [c70]	●			●			U	s	Dempster-Shafer
Litvak et al. [c51]			●	●			U	b	BVUML
Rasch, Wehrheim [c64]	●		●				U	s/b	Z, CSP/FDR
* Wirsing, Knapp [c75]	●		●	●			T	s/b	univ. alg.
Kyas et al. [c47]	●		●			●	U	s/b	PVS
van der Straeten [c71, c72]	●		●	●			U	s	Desc. Logic
Amálio et al. [c2]	●	●	●				U	s	Z
Kim, Carrington [c41]	●		●				U	s	Object-Z
Diethers, Huhn [c16]			●	●			U	b	UPPAAL
Yang et al. [c76]	●			●			U	s	rCOS
Yeung [c78]	●		●				U	b	CSP, B

Table 1. Overview of UML/OCL 1 consistency approaches. CD means class diagrams, OD object diagrams, SM state machines, ID interaction diagrams (e.g. sequence diagrams), AD activity diagrams (as a special case of state machines), and OCL the Object Constraint Language. A ● means support for at least a substantial subset of the diagram/sub-language type, a ● indicates that the diagram/sub-language is supported but only for a limited subset. The consistency technique of the approach is indicated by an S for “system model”, D for “dynamic meta-modelling”, U for “universal logic”, or T for “heterogeneous transformation”. An entry “s” in the class(ification) column means that structural, static consistency checks are supported, a “b” that behavioural, dynamic consistency is checked; if the indicator is set into parentheses, the consistency support is quite restricted. The last column shows the used formalisms and tools. An asterisk in front of the reference indicates that more information is given in Sect. 2.1.

Reference	CD	OD	CMP	CSD	SM	ID	AD	OCL	cons.	class.	Form./Tool
Lam, Padget [c49]					●	●			U	b	π -calculus
Long et al. [c54]	●						●		U	s	rCOS
Lucas et al. [c55]	●						●		U	s	Maude
Okalas et al. [c61]	●					●			U	s/b	B
Rasch, Wehrheim [c65]	●		●		●	●			U	s/b	Z, CSP/FDR
Wang et al. [c74]						●	●		U	b	LTSA
Bellur, Vallieswaran [c6]	●		●		●	●			U	s	meta-model
Li et al. [c50, c53]	●	●			●	●			U	s/(b)	UTP
O'Keefe [c62]	●				●	●			U	b	Dynamic Logic
Shinkawa [c68]	●				●	●	●		U	b	CPN
Yao, Shatz [c77]					●	●			U	b	Petri nets
Zhao et al. [c79]					●	●			U	b	SPIN
Anastasakis et al. [c3]	●							●	U	s/(b)	Alloy
* Gogolla et al. [c29]	●	●			●	●		●	U	s/(b)	USE
Knapp, Wuttke [c43]	●				●	●			U	b	Hugo/RT
Sapna, Mohanty [c67]	●				●	●	●		U	s	SQL
Brændshøi [c8]					●	●			U	b	impl.
Banerjee et al. [c4, c5]	●				●				U	b	Rhapsody/LTL
* Cengarle et al. [c11]	●					●		●	T	s/b	institutions
* Alanazi [c1]					●	●			U	(b)	impl.
Hammal [c34]					●	●			U	(b)	Petri nets
Laleau, Polack [c48]	●				●	●			U	s	meta-model
* Broy et al. [c9, c10]	●	●			●	●			S	s/b	set theory
* Kuske et al. [c44]	●	●			●	●			U	(s/b)	graph. transf.
* Grönniger [c31]	●	●			●	●		●	S	s/b	Isabelle/HOL
Nimiya et al. [c58]					●	●			U	b	Alloy
Khai [c39]	●						●		U	s	Prolog
Ober, Dragomir [c59]			●	●	●				U	s/b	OMEGA2
Puczyński [c63]	●				●	●			U	s/b	impl.
Gerlinger et al. [c28]	●			●			●		U	s	Common Logic
El Miloudi et al. [c21, c22]	●				●	●			U	s	Z
Khan, Porres [c40]	●	●			●			●	U	s	Desc. Logic
* fUML [c60]	●						●		S	s/b	Common Logic

Table 2. Overview of UML/OCL 2 consistency approaches. The abbreviations are as in Tab. 1 extended by CMP for component diagrams and CSD for composite structure diagrams. Activity diagrams (AD) have an independent semantics in UML 2. Protocol state machines are not equipped with a diagram type of their own in UML 2; still, [c29] considers them independently.

20, 28, 41, 42], the information of which we aligned, adapted, and extended by search queries on the Internet and personal experiences.

From the 14 different UML diagram types (structural: profile, class, composite structure, component, deployment, object, package; behavioural: activity, sequence, communication, interaction overview, timing, use case, state machine [34, p. 681]), we combined, as usual, the sequence, communication, interaction overview and timing diagram into the single type of interaction diagram for conciseness; and we omitted the profile, deployment, package, and use case diagram. Package diagrams provide a

means for namespace modularisation and the package structure may most of the time be resolved statically using fully qualified names; still, also the meaning of packages and their relationships has been discussed [9, 38]. Use case diagrams, though besides class diagrams the most used diagram type of the UML [10, 26], convey rather little semantics on their own [19, 23]. Deployment diagrams, assigning software artefacts to system elements, also show quite limited semantic content (but cf. [30]). Finally, profile diagrams are used to define a domain-specific UML extension, thus every instance would add a viewpoint of its own.

Our survey thus covers 11 sub-languages of the UML/OCL family, where the entry for interactions condenses the information on four diagram sub-types. Since we aim at multi-view consistency involving several viewpoints, we do not list approaches here that only consider a single diagram type. In particular, we leave out the consistency of class diagrams, possibly accompanied by object diagrams (see [6] for an overview) or state machines [27, 36, 37]. Class diagrams, however, have been, in fact, the first instance of UML consistency investigations [13, 14, 24].

We now first review the variety of techniques enabling consistency checking in the listed multi-view approaches, and then report on our general observations and findings.

2.1 Consistency Techniques

The most immediate and direct approach to consistency checking of UML diagrams and models uses consistency rules, mostly on the concrete or abstract syntax. These rules extend the well-formedness rules of the UML specification given in OCL [c13]. Another option for such rules is to use other kinds of logics, like description logics [c40, c71, c72]. Many modelling tools incorporate their own rule sets [c17, c20, c52, c73]. The survey by Torre et al. [41] lists 116 consistency rules studied in the literature, where 95 are syntactic (structural).

Syntactic checks are indispensable in any approach to consistency, but they do not suffice to uncover the more intricate behavioural consistency problems, e.g., whether a network of state machines admits a trace specified by an interaction. Advanced consistency approaches thus have to develop and rely on a behavioural semantics of the UML/OCL diagrams and sub-languages of discourse. The degree of integration of these semantics varies considerably with the proposed approaches in the literature. Though the borders can not be always drawn with full accuracy, we suggest a categorisation w.r.t. the emphasis which is given to the semantic heterogeneity of UML/OCL. In the “system model” approach a uniform realisation frame is built, into which all sub-language aspects are encoded. The “dynamic meta-modelling” approach dispenses with the encoding, but enriches the meta-model, i.e., the abstract syntax of the UML, by semantic information. The “universal logic” approaches still use an encoding, though now to a uniform formalism. Finally, “heterogeneous transformations” approaches aim at employing families of translations for relating sub-languages.

System Model. The “system model” approach, best exemplified by Broy, Groenniger et al. [c9, c10, c31], builds on a uniform semantic basis for covering all aspects of state and state change present in any UML sub-language to be considered. By representing every facet of a model, expressed in various diagrams, in one and the same instance of

the system model, static as well as dynamic checks can be performed. The “Executable UML” (xUML [c57]) as well as the “Foundational Subset of the UML” (fUML [c60]) use such system models for comprehensive and integrated execution.

For states, the system model in [c9, c10], contains a data store built from classes, their attributes, and the inheritance relationship as well as the instances; a control store consisting of operations and stacked method calls; and an event store holding also messages. For state changes, it comprises control-flow and event-based state transitions systems enriched by time. This system model, though with some modifications, e.g., specialising the event store to a message store, has later on been encoded in Isabelle/HOL and parts of UML class and object diagrams, state machines, and sequence diagrams, as well as a subset of OCL have been represented in this system model [c31]. With the help of the Isabelle prover then both static consistency checks, like whether an inheritance relationship is acyclic, and dynamic consistency checks, like whether a sequence diagrams is realisable by a state machine, can be done.

The manual effort to write down these checks and perform them in an interactive theorem prover seems quite substantial, however. Not to the least part, this is owed to the necessary complexity of the system model. Automation of various consistency checks has not been the primary goal of the approach. Still, the approach supports a certain degree of variability by exchanging sub-theories of the encoded system model, and other languages, a programming language, for instance, can be integrated [c31] if they can also be represented adequately in the system model.

The “krtUML” approach by Damm et al. [c14] uses symbolic transition systems as its system model for a comprehensive semantics. Their choice of class diagrams and state machines targets real-time systems. Consistency checks are not the main goal but behavioural consistency may be added on the basis of this system model. They stress that “[b]ecause all diagrams are only views on one and the same model, the attempts to give semantics for separated UML diagrams fail in producing the right semantics for the entire UML” (p. 94), though this valid observation somewhat neglects the possibility to integrate the relations between the UML diagrams and sub-languages as done, e.g., in the heterogeneous transformation approaches.

Dynamic Meta-modelling. Inspired by attribute grammars that extend the abstract syntax tree of a (textual) language by synthesised and inherited attributes for semantic and contextual analysis, the “dynamic meta-modelling” approach by Hausmann, Engels, et al. [c12, c24, c25, c27, c35, c69] extends the abstract syntax of the UML, i.e., its meta-model, by semantic concepts on this very meta level. In fact, the UML meta-model shows several concepts that serve as links between the various sub-languages, like Event originating from, e.g., operation calls, used in state machines and activities for triggering behavioural effects or Message used in interactions for referring to operations and signals. By adding extra semantic concepts, the linkage between the sub-languages can be enhanced and, in particular, lifted to the behavioural, dynamic level. For example, the UML meta-class `StateMachine` is extended by a new meta-class `EventPool` for holding instances of the already existing meta-class `Event` that the instance of `StateMachine` then can react to; or the meta-class `ControlFlow` of activities is extended by a new meta-class `ControlToken` representing the possibility that a control flow activity edge may carry a control token. Using the extensions, an operational semantics based on the extended

meta-model and thus covering several UML sub-languages in concert is defined using (typed) graph transformations with negative application conditions, most prominently in the GROOVE tool [c35, c69] applying state space exploration. The graph transformation rules are separated into local small-step rules and transactional big-step rules that call the local rules.

This intriguing idea, combining attribute grammars and structural operational semantics, has mainly been applied to activities [c35] and to a limited degree to state machines [c69] and OCL [c12]. For consistency checks proper, the dynamic relation between sequence diagrams and state machines has been considered as an example, though without tool support [c25]. The overall design of dynamic meta-modelling somewhat resembles the “system model” approach as it builds a single domain of interpretation. By contrast, however, dynamic meta-modelling does not rely on an external semantic domain, but reuses the existing UML concepts and adds those features directly to the meta-model that are missing for behavioural interpretation. The use of a reference model for relating views that are then embedded and integrated into system model has already been advocated by Ehrig et al. [12]; the use of graph transformations on the meta-model level has also been used by Kuske et al. [c44]. Still, the complexity of the UML meta-model itself, let alone the necessary extensions, and respecting all semantic interconnections in local and global graph transformation rules present a major obstacle for the use of dynamic meta-modelling in comprehensive multi-view consistency checking.

Universal Logic. The system model approach builds a uniform semantic domain offering the necessary mechanism to interpret the different UML sub-languages and diagrams. By contrast, a “universal logic” approach does not rely on a single domain of interpretation, but just uses a uniform logical technique, like transition systems, for expressing the semantics of all UML diagrams to be checked for consistency. However, as in the system model approach, having to use a single encoding technique may sometimes yield unnecessary and unnatural complexity.

Große-Rhode [c32, c33] uses “transformation systems”, i.e., extended labelled transition systems, where both states and transitions are labelled. The (control) states offer observations and synchronisation points, the transitions model the atomic steps of an entity and may be executed synchronously with other system parts. The semantics of class diagrams, state machines as well as their composition, and sequence diagrams are represented as classes of such transformation systems. Consistency can then be expressed by checking that the intersection of model classes (modulo some projections for adapting labels) are not empty. In a similar vein, though not as elaborated, the “super-state analysis” of Alanazi [c1] relies on nets of transition systems which allows to check the consistency of state machines and interactions.

The “Consistency Workbench” by Küster, Engels, et al. [c23, c26, c45] is based on a “partial translation of models into a formal language (called semantic domain) that provides a language and tool support to formulate and verify consistency conditions” [c23, p. 158]. In principle, the employed semantic domain is not fixed for all installments of the general approach and may vary with the consistency checks and the information extracted from the models by partial translation. The Consistency Workbench itself relies exclusively on the algebraic process language CSP and failure-divergence refinement (FDR). Still, it does not aim to construct an overarching system model, but is param-

eterised in the consistency problem type. In this sense, it is bordering at an approach using heterogeneous transformation.

In the “film-stripping” approach by Gogolla et al. [c30, c36] a uniform technique for representing behavioural system evolution is used: System behaviour is captured by sequences of snapshots of system states, i.e., object diagrams, linked together by change information in particular recording how the objects evolve. Consistency checks could then be performed, e.g., in the USE tool [c29]. Even for automated analysis, like model checking, however, the general scaling of the technique without appropriate compression or abstraction of the snapshots remains unclear. The approach is complemented by model transformations from full-fledged UML to a simpler “base model” [c37, c38]: Complex modelling constructs, like association classes, are replaced with simpler modelling expressions, though possibly at the expense of having to use OCL. This technique is mainly exemplified by transformations on class diagrams and OCL itself, though, ultimately “[a]ll diagrams conjoined are transformed and combined into a base model” [c38, p. 60]. Thus, there are quite some similarities with the system model approach.

Heterogeneous Transformation. Approaches based on “heterogeneous transformation” (coined by [c19]) focus on the several sub-languages and diagrams of the UML used in different forms at different development stages, from different viewpoints by different stakeholders [21, 22]. Such an approach has in particular been advocated by Derrick et al. [c15] for UML from their experiences with “Open Distributed Processing” (ODP) using Z and LOTOS, though not elaborated in detail. In their terminology a set of viewpoint specifications is consistent “if there exists a specification that is a refinement of each of the viewpoint specifications with respect to the identified refinement relations and the correspondences between viewpoints” (p. 35).

Egyed’s VIEWINTEGRA [c18, c19] defines transformations between the different UML diagram types on the very diagram level. In principle, all eleven diagram types of UML 1 could be covered, but only class diagrams, object diagrams, state machines, and interaction diagrams, i.e., sequence and collaboration diagrams (which, as in UML 2, are mere visual variants), are discussed in [c19]. The transformations are categorised into generalisation, e.g., object to class diagram; structuralisation, e.g., state machine to class diagram; translation, e.g., sequence into collaboration diagram; and abstraction, e.g., class diagram to class diagram. The last class of transformations, abstraction, is employed to relate diagrams and different development and refinement stages. Since the transformations map diagrams to diagrams, the supported consistency checks are structural; neither a static nor a dynamic semantics are provided. The classification of transformations is also used to reduce the number of necessary comparison transformations, which for eleven diagram types would otherwise be 55. However, when employing this design, not all transformations are possible any more, and a common denominator sometimes is needed, e.g., for comparing an object diagram with a state machine both are, perhaps somewhat arbitrarily, transformed into a class diagram.

An effort to formalise the relation of views and viewpoints on a semantic level is provided in [c75], where viewpoints are captured by a formal language category equipped with a model functor to a semantic domain category and views are language expressions in a viewpoint. Consistency is expressed by translations on the syntactic and semantic level; for semantic consistency a “viewpoint of comparison” has to be given.

Consistency checks, though only pair-wise, are exemplified for class diagrams, state machines, and sequence diagrams. The scheme by Cengarle et al. [c11] is similar, but uses the established theory of institutions as its foundation.

2.2 Observations and Results

Not all 53 approaches listed in Tabs. 1 and 2 are of the same quality w.r.t. elaboration and comprehensiveness with a kind of feasibility study [c34, c58] and detailed accounts involving comprehensive semantics, tools, and larger case studies [c29, c43, c47, c59] as the ends. Tool support is similarly diverse and ranges from prototypical proprietary implementation over model checking and model finders to interactive theorem proving. In line with the survey results by Torre et al. [40], we also find that the diagram types most often covered are the class diagram (40 out of 53), the state machine diagram (44), and the interaction diagram (in one of its forms, 38); that according to our survey state machines have been considered more frequently than class diagrams may be accounted by our judicious choice of omitting single-view consistency approaches. Activity diagrams have rarely been integrated (5), but this may change with the broader adoption of fUML [c60]. The combination of state machines and interactions is considered most often (32). They are mainly considered in the context of class diagrams, rarely in combination with component diagrams and composite structure diagrams (3).

Among the approaches listed covering many different diagram types, the top four are the following ones:

- xUML [c57] covers five diagram types. However, for three of them, only a limited subset is covered, and only syntactic, structural checks are provided.
- Gogolla et al. [c29] cover six diagram types, three of them to a substantial portion, and at least partially semantic, behavioural checks are provided.
- Broy et al. [c9, c10] define a comprehensive, though complex system model capturing substantial subsets of four diagram types of UML; however, tool support is not provided.
- Grönniger [c31], based on [c9, c10], integrates a fragment of the UML (partially covering class diagrams, object diagrams, interactions, state machines, and OCL) semantically by an encoding to the interactive theorem prover Isabelle/HOL.

All these either follow the system model or the universal logic approach. In accordance with the survey by Lucas et al. [28] we also find, that these encoding techniques (system model: 6, universal logic: 43) are currently by far prevailing for consistency. One major drawback of the encoding approaches, which may have prevented the further integration of other UML/OCL diagram types and sub-languages into them, is their lack of extensibility: A new viewpoint really extending the realm of expressivity inevitably calls for a considerable amount of work in expanding and adapting the already established semantic or logical domain.

3 Distributed Semantics for Multi-view Consistency

The previous section has shown that we are still quite far away from a comprehensive approach to multi-view consistency for UML. One reason is the complexity inherent in

the diversity of UML models and their interaction. In [31], we have distinguished three different approaches to handle semantic heterogeneity:

- encoding of heterogeneous languages into some “universal” language. This approach is applied by both the system model, the dynamic meta-modelling and the universal logic approaches described in Sect. 2;
- focused heterogeneity, i.e. languages are not per se encoded into some “universal” language, and complex models may involve parts written in different languages. Still, via translations, the end result is formulated in one language. This roughly corresponds to the heterogeneous transformation approach described in Sect. 2;
- distributed heterogeneity, i.e. truly decentralised networks of models formulated in different languages. This is a *heterogeneous transformation approach* that we propose to use.

Note that the existing heterogeneous transformation approaches are *not* distributed: they fall short in leveraging the translations to build up a distributed system of viewpoints. Therefore, they fail to handle the complexity of multi-view models in a comprehensive way. Such a comprehensive *and* semantic treatment is only possible with a decentralised approach, namely the distributed heterogeneous transformation approach. This means the provision of formal models for the involved UML/OCL diagram types and sub-languages that directly follow the intended semantics as specified in the UML/OCL standards. Here, the usual semi-formal language of mathematics is used, instead of casting the formal model into some predefined formal language. For a semantics distributed heterogeneity, these different formal models have to be linked, of course.

3.1 An institutional approach to distributed heterogeneous transformation

A useful meta notion for carrying out this formalisation is the notion of institution [15], an abstract formalisation of the notion of logical system. See [c56] for an early mentioning of institutions in the context of UML. The integration of different UML/OCL sub-languages (class diagrams, OCL, interactions) formalised as institutions has been started by Cengarle et al. [c11]. We have sketched the extension to further UML diagram types and a general vision in [c42].

A central feature of institutions is the provision of a notion of *realisation* of a model. Note that in institution-theoretic terms, this is called a model (in the sense of model theory) of a logical theory. However, this terminology can lead to confusion in the domain of model-driven engineering (MDE), where MDE models play the roles of logical theories. That is why we prefer the term “realisation” (of an MDE model) here. [18] speak of instantiation; however, this is tailored towards class diagrams. Realisations of models also exist for state machines (these are certain transition systems), sequence diagrams (there, they are trace sets), etc. They can differ greatly from institution to institution. The central point is that the notion of realisation provides a notion of *consistency*: a (single-view) model is consistent iff it has at least one realisation. Thus, the different notions of consistency from [18] can be captured by using different institutions. Moreover, these notions of realisation and consistency can be extended from the horizontal (or intra-model) case (i.e., living within one sub-language resp. institution) to the vertical (or inter-model) case of multi-view models, using so-called *networks*, see below.

We here describe general methods how to address the problem of multi-view consistency in this setting. Central tool is the Distributed Ontology, Model and Specification Language (DOL)⁴, which recently has been adopted as a standard by the Object Management Group (OMG). DOL is not yet another modeling language, but rather a meta language for the specification of relation between different existing models. That is, UML diagrams can be referenced in DOL as-is, without the need of an encoding into some other language. The only need is to specify a formal semantics for the involved UML diagram types in the form of an institution. For UML class diagrams, this has been done in an informative appendix of the DOL standard itself.⁵ For other UML diagram types, this has been partially done, see [c42] for an overview.

Translations between institutions can be formalised as so-called institution morphisms and comorphisms [16]. An institution morphism roughly corresponds to a projection from a “richer” to a “poorer” logic, expressing that the “richer” logic has some more features, which are forgotten by the morphism. The main purpose of the institution morphisms is the ability to express, e.g., that an interaction diagram and a state machine are compatible because they are expressed over the same class diagram. By contrast, institution comorphisms are often more complex. Roughly, a comorphism corresponds to an encoding of one logic into another one.

We now illustrate this framework with a few DOL examples.

3.2 ATM Example

In order to illustrate our approach to a heterogeneous institutions-based UML semantics in general and the institutions for UML state machines in particular, we use as a small example the design of a traditional automatic teller machine (ATM) connected to a bank. For simplicity, we only describe the handling of entering a card and a PIN with the ATM. After entering the card, one has three trials for entering the correct PIN (which is checked by the bank). After three unsuccessful trials the card is kept.

Let us assume that an institution morphism `sd2cd` from UML sequence diagrams to UML class diagrams has been defined. It extracts the classes, attributes, operations, signals etc. used in the sequence diagram and forms them into a class diagram. Given a sequence diagram `ATM_Bank_Interaction`, the DOL declaration for extracting its underlying class diagram is then

```
model ATM_Bank_Interaction_cd =
  ATM_Bank_Interaction hide along sd2cd
end
```

We now can express that a class diagram `User_Interface` refines to the sequence diagram `ATM_Bank_Interaction` in DOL:

```
refinement r1 =
  { User_Interface reveal ATM_Bank_Interaction_cd }
  refined to ATM_Bank_Interaction_cd
```

⁴ <http://dol-omg.org>

⁵ This is the first semantics of UML class diagrams that has been reviewed by co-designers of UML.

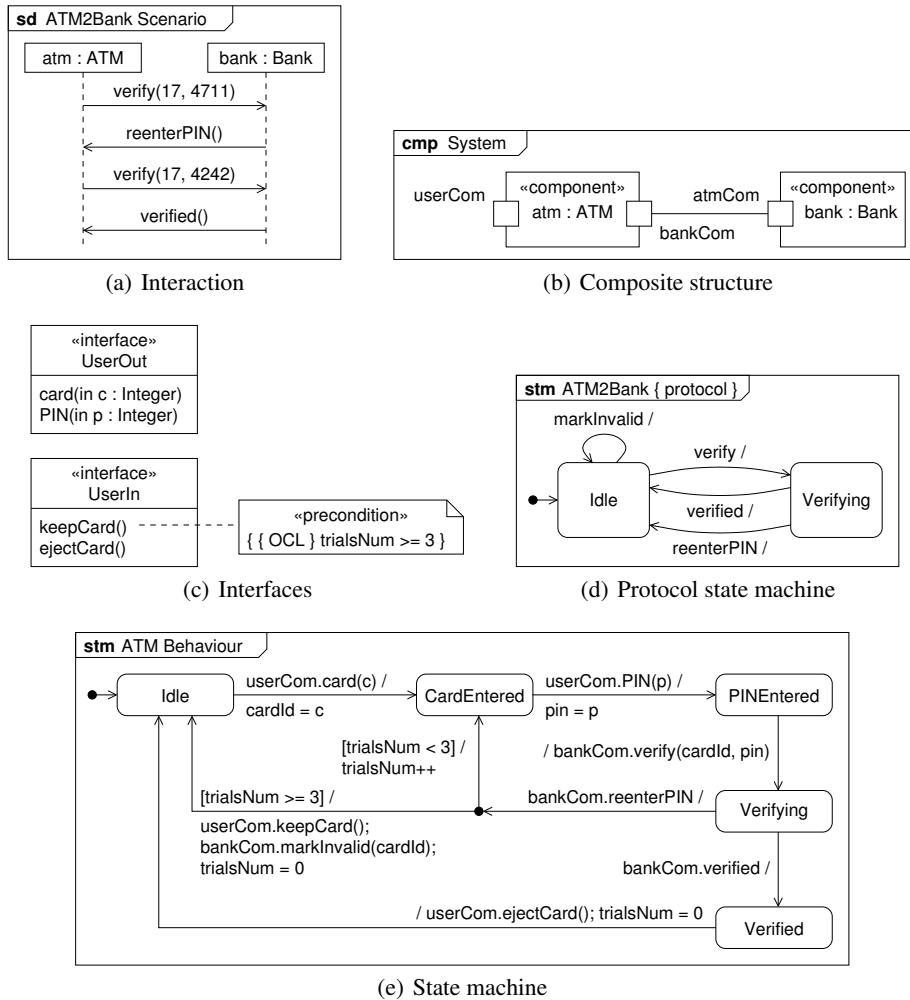


Fig. 1. ATM example

end

The semantics of such a refinement is inclusion of realisation. In this case, it means that each realisation of `ATM_Bank_Interaction_cd` is also a realisation of

```
{ User_Interface reveal ATM_Bank_Interaction_cd }
```

More specifically, this mean that each realisation of `ATM_Bank_Interaction` (which is a set of traces), when organised into a transition system of snapshots, is a transition system of the class diagram `User_Interface` (when the latter is restricted to those symbols that actually occur in `ATM_Bank_Interaction`).

Using a comorphism `cd2stm`, we can express that a state machine is built over some class diagram as follows:

```

model ATM_stm =
  User_Interface with translation cd2stm
then
  ATM_stm_definition
end

```

Now suppose that we have a second state machine (over the same class diagram, but typically not associated to the same class):

```

model Bank_stm =
  User_Interface with translation cd2stm
then
  Bank_stm_definition
end

```

and we have a system of these two state machines linked by a composite structure diagram within a component that we call *System*. This could be expressed as follows, using a comorphism *stm2cmp* linking state machines and composite structure diagrams:

```

model System =
  ATM_stm with translation stm2cmp with cid |-> atm
and
  Bank_stm with translation stm2cmp with cid |-> bank
then
  cmp
end

```

We now express that this system can realise the interactions expressed in sequence diagram *ATM_Bank_Interaction* as follows:

```

refinement r2 =
  ATM_Bank_Interaction refined to { System hide along cmp2sd }
end

```

Multi-view models can be expressed as so-called *networks* in DOL. A network consists of a number of component models (the views), plus some links (e.g. refinements) between these. So we obtain a network of the above UML diagrams and state its consistency as follows:

```

network N = %consistent
  User_Interface, ATM_stm, Bank_stm, System,
  ATM_Bank_Interaction, r1, r2
end

```

A realisation of the network consists of a family of realisations for each of the different components (views) of the network that is *compatible* along the links.⁶ Consistency of the network means existence of at least one realisation. Note that consistency of a network is more than pairwise consistency of each pair of models involved.

⁶ See [31] and the DOL standard at <http://dol-omg.org> for details.

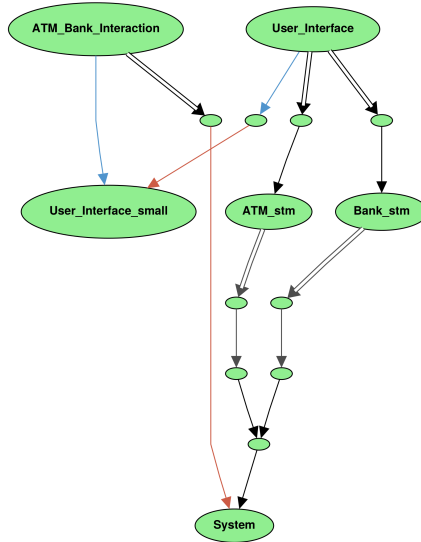


Fig. 2. Development graph for the ATM/Bank system

An important question is how to check consistency of such a heterogeneous multi-view network. One possibility follows the system model paradigm: using comorphisms, all involved models are translated into one formalism (e.g. first-order logic or Isabelle/HOL), and then consistency is checked there. This amounts to checking the consistency of a (homogeneous) logical theory, which can be done using standard realisation finders⁷.

A second option is more decentralised and starts with separate realisations of all the involved models. Such realisations generally can be expressed in DOL: for example, for a class diagram, a realisation can be expressed as an object diagram. For a state machine, a so-called free DOL expression can be used to specify a particular DOL realisation. In order to check that such a family of realisations is a realisation of the network, pairwise compatibility of these realisations needs to be checked. Of course, this will generally fail if the individual realisations have been randomly produced by a realisation finder. This means that the realisations need to be carefully designed in a manual way having in mind that they must compose into a realisation of the overall network.

This decentralised approach of finding a realisation is largely manual. More automation can be obtained using an incremental approach. Starting with class diagrams, OCL expressions and object diagrams, realisations for the static parts are obtained, e.g. using a realisation finder for first-order logic. In the next step composite structure diagrams and state machines are considered, and then sequence diagrams.

⁷ Usually, these are called model finders, e.g. for first-order logic. However, in our terminology, a first-order model of a logical theory would be a realisation of (the translation) of an MDE model.

The use of realisation finders needed for consistency checking in all these approaches quickly comes to its limits when large models (or logical theories, after e.g. translation to first-order logic) are involved. In [25], we have developed an approach of decomposing large consistency problems into smaller ones, using so-called CASL architectural specifications.

4 Conclusion and Future work

UML/OCL is a language for multi-view and multi-viewpoint models, and the detection of view consistency at an early stage of the development is important for avoiding costly redesign. We have classified 53 existing approaches to UML/OCL multi-view consistency. Even the best approaches cover only five of the 14 UML diagram types, and most of these only partially. Moreover, a “universal logic” approach is predominant, where all UML/OCL diagram types and sub-languages are embedded into one system model or one logic. We have argued that this is not suitable for handling the involved complexity.

We propose a new approach to UML multi-view consistency, following a “heterogeneous transformation” paradigm. We use institutions for formalising the different UML diagram types and their semantics, and institution (co-)morphisms for formalising the transformations. Then UML multi-viewpoint models can be formalised as so-called networks in the OMG-standardised Distributed Ontology, Model, and Specification Language (DOL). This provides a framework where eventually all semantically relevant diagram types can be covered.

In order to use this framework for checking consistency of UML multi-viewpoint models, there is still a considerable way to go: while some UML diagram types have been formalised as institutions, this needs to be completed to a more comprehensive treatment of both diagram types and their features. Formalisation of transformations as institution (co-)morphisms has just started. In order to make this practically useful in connection with DOL, all these institutions and (co-)morphisms need to be integrated into the Heterogeneous Tool Set (Hets) and interfaced with suitable proof and model finding tools. Finally, suitable consistency strategies need to be developed and implemented.

Of course, writing down DOL expressions for large families of UML diagrams will be tedious. Hence, we aim at some graphical interface that can generate the needed DOL expressions automatically from a user’s selection of those UML diagrams that should be interlinked to a network, plus a specification of the involved refinements. Such a specification of both networks and refinements adds the extra information to a given family of UML diagrams that is needed when checking multi-view consistency.

Multi-view UML Consistency Approaches

- c1. M. N. Alanazi. “Consistency Checking in Multiple UML State Diagrams Using Super State Analysis”. PhD thesis. Kansas State University, 2008.
- c2. N. Amálio, S. Stepney, and F. Polack. “Formal Proof from UML Models”. In: *Proc. 6th Intl. Conf. Formal Methods and Software Engineering (ICFEM’04)*. Ed. by J. Davies, W. Schulte, and M. Barnett. Lect. Notes Comp. Sci. 3308. Springer, 2004, pp. 418–433.

- c3. K. Anastasakis, B. Bordbar, G. Georg, and I. Ray. “UML2Alloy: A Challenging Model Transformation”. In: *Proc. 10th Intl. Conf. Model Driven Engineering Languages and Systems (MoDELS’07)*. Ed. by G. Engels, B. Opdyke, D. C. Schmidt, and F. Weil. Lect. Notes Comp. Sci. 4735. Springer, 2007, pp. 436–450.
- c4. A. Banerjee, S. Ray, P. Dasgupta, P. P. Chakrabarti, S. Ramesh, and P. V. V. Ganesan. “A Dynamic Assertion-based Verification Platform for Validation of UML Designs”. In: *Proc. 6th Intl. Symp. Automated Technology for Verification and Analysis (ATVA’08)*. Ed. by S. Cha, J.-Y. Choi, M. Kim, I. Lee, and M. Viswanathan. Lect. Notes Comp. Sci. 5311. Springer, 2008, pp. 222–227.
- c5. A. Banerjee, S. Ray, P. Dasgupta, P. P. Chakrabarti, S. Ramesh, and P. V. V. Ganesan. “A Dynamic Assertion-based Verification Platform for Validation of UML Designs”. In: *ACM SIGSOFT Softw. Eng. Notes* 37.1 (2012), pp. 1–14.
- c6. U. Bellur and V. Vallieswaran. “On OO Design Consistency in Iterative Development”. In: *Proc. 3rd Intl. Conf. Information Technology: New Generations (ITNG’06)*. IEEE, 2006, pp. 46–51.
- c7. S. Bernardi, S. Donatelli, and J. Merseguer. “From UML Sequence Diagrams and Statecharts to Analysable Petri Net Models”. In: *Proc. 3rd Intl. Ws. Software and Performance (WSOP’02)*. ACM, 2002, pp. 35–45.
- c8. B. Brændshøi. “Consistency Checking UML Interactions and State Machines”. Master thesis. Universitetet i Oslo, 2008. URL: <https://www.duo.uio.no/bitstream/handle/10852/9992/bjornbra.pdf>.
- c9. M. Broy, M. V. Cengarle, H. Grönniger, and B. Rumpe. “Considerations and Rationale for a UML System Model”. In: *UML 2 — Semantics and Applications*. Ed. by K. Lano. Wiley, 2009. Chap. 3, pp. 43–60.
- c10. M. Broy, M. V. Cengarle, H. Grönniger, and B. Rumpe. “Definition of the System Model”. In: *UML 2 — Semantics and Applications*. Ed. by K. Lano. Wiley, 2009. Chap. 4, pp. 61–93.
- c11. M. V. Cengarle, A. Knapp, A. Tarlecki, and M. Wirsing. “A Heterogeneous Approach to UML Semantics”. In: *Concurrency, Graphs and Models, Essays Dedicated to Ugo Montanari on the Occasion of His 65th Birthday*. Ed. by P. Degano, R. De Nicola, and J. Meseguer. Lect. Notes Comp. Sci. 5065. Springer, 2008, pp. 383–402.
- c12. J. M. Chiaradía and C. Pons. “Improving the OCL Semantics Definition by Applying Dynamic Meta Modeling and Design Patterns”. In: *Proc. 6th OCL Ws. (OCL’06). OCL for (Meta-)Models in Multiple Application Domains*. Ed. by D. Chiorean, B. Demuth, M. Gogolla, and J. Warmer. Electr. Comm. EASST 5. 2006.
- c13. D. Chiorean, M. Paşca, A. Cârçu, C. Botiza, and S. Moldovan. “Ensuring UML Models Consistency Using the OCL Environment”. In: *Proc. 3rd OCL Ws. (OCL’03). OCL 2.0*. Ed. by P. H. Schmitt. Electr. Notes Theo. Comp. Sci. 102. Elsevier, 2004, pp. 99–110.
- c14. W. Damm, B. Josko, A. Pnueli, and A. Votintseva. “Understanding UML: A Formal Semantics of Concurrency and Communication in Real-Time UML”. In: *Rev. Lect. 1st Intl. Symp. Formal Methods for Components and Objects (FMCO’02)*. Ed. by F. S. de Boer, M. M. Bonsangue, S. Graf, and W.-P. de Roever. Lect. Notes Comp. Sci. 2852. Springer, 2003, pp. 71–98.
- c15. J. Derrick, D. Akehurst, and E. Boiten. “A Framework for UML Consistency”. In: *Proc. Ws. Consistency Problems in UML-based Software Development*. 2002, pp. 30–45.
- c16. K. Diethers and M. Huhn. “Vooduu: Verification of Object-Oriented Designs Using UP-PAAL”. In: *Proc. 10th Intl. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS’04)*. Ed. by K. Jensen and A. Podelski. Lect. Notes Comp. Sci. 2988. Springer, 2004, pp. 139–143.
- c17. R. Dubauskaite and O. Vasilecas. “Method on Specifying Consistency Rules among Different Aspect Models, expressed in UML”. In: *Elektr. Elektrotechn.* 19.3 (2013), pp. 77–81.

- c18. A. Egyed. “Heterogenous View Integration and its Automation”. PhD thesis. University of Southern California, 2000.
- c19. A. Egyed. “Scalable Consistency Checking between Diagrams — The VIEWINTEGRA Approach”. In: *Proc. 16th IEEE Intl. Conf. Automated Software Engineering (ASE’01)*. IEEE, 2001, pp. 387–390.
- c20. A. Egyed. “UML/Analyzer: A Tool for the Instant Consistency Checking of UML Models”. In: *Proc. 29th Intl. Conf. Software Engineering (ICSE’07)*. IEEE, 2007, pp. 793–796.
- c21. K. El Miloudi, Y. E. Amrani, and A. Ettouhami. “An Automated Translation of UML Class Diagrams into a Formal Specification to Detect UML Inconsistencies”. In: *Proc. 6th Intl. Conf. Software Engineering Advances (ICSEA’11)*. 2011, pp. 432–438.
- c22. K. El Miloudi and A. Ettouhami. “A Multi-View Approach for Formalizing UML State Machine Diagrams Using Z Notation”. In: *WSEAS Trans. Comp.* 14 (2015), pp. 72–78.
- c23. G. Engels, R. Heckel, and J. M. Küster. “The Consistency Workbench: A Tool for Consistency Management in UML-Based Development”. In: *Proc. 6th Intl. Conf. Unified Modeling Language (UML’03)*. Ed. by P. Stevens, J. Whittle, and G. Booch. Lect. Notes Comp. Sci. 2863. Springer, 2003, pp. 356–359.
- c24. G. Engels, J. H. Hausmann, R. Heckel, and S. Sauer. “Dynamic Meta Modeling: A Graphical Approach to the Operational Semantics of Behavioral Diagrams in UML”. In: *Proc. 3rd Intl. Conf. Unified Modeling Language (UML’00)*. Ed. by A. Evans, S. Kent, and B. Selic. Lect. Notes Comp. Sci. 1939. Springer, 2000, pp. 323–337.
- c25. G. Engels, J. H. Hausmann, R. Heckel, and S. Sauer. “Testing the Consistency of Dynamic UML Diagrams”. In: *Proc. 6th World Conf. Integrated Design & Process Technology (IDPT’02)*. 2002.
- c26. G. Engels, R. Heckel, and J. M. Küster. “Rule-based Specification of Behavioral Consistency Based on the UML Meta-model”. In: *Proc. 3rd Intl. Conf. Unified Modeling Language (UML’01)*. Ed. by M. Gogolla and C. Kobryn. Lect. Notes Comp. Sci. 2185. Springer, 2001, pp. 272–286.
- c27. G. Engels, C. Soltenborn, and H. Wehrheim. “Analysis of UML Activities Using Dynamic Meta Modeling”. In: *Proc. 9th IFIP WG 6.1 Intl. Conf. Formal Methods for Open Object-Based Distributed Systems (FMODS’07)*. Ed. by M. M. Bonsangue and E. B. Johnsen. Lect. Notes Comp. Sci. 4468. Springer, 2007, pp. 76–90.
- c28. A. Gerlinger Romero, K. Schneider, and M. Gonçalves Vieira Ferreira. “Integrating UML Composite Structures and fUML”. In: *Proc. 40th Conf. Current Trends in Theory and Practice of Computer Science (SOFSEM’14)*. Ed. by V. Geffert, B. Preneel, B. Rován, J. Štuller, and A. M. Tjoa. Lect. Notes Comp. Sci. 8327. Springer, 2014, pp. 269–280.
- c29. M. Gogolla, F. Büttner, and M. Richters. “USE: A UML-based Specification Environment for Validating UML and OCL”. In: *Sci. Comput. Program.* 69.1-3 (2007), pp. 27–34.
- c30. M. Gogolla, L. Hamann, F. Hilken, M. Kuhlmann, and R. B. France. “From Application Models to Filmstrip Models: An Approach to Automatic Validation of Model Dynamics”. In: *Proc. Modellierung 2014*. Ed. by H.-G. Fill, D. Karagiannis, and U. Reimer. Lect. Notes Inf. 225. GI, 2014, pp. 273–288.
- c31. H. Grönniger. “Systemmodell-basierte Definition objektbasierter Modellierungssprachen mit semantischen Variationspunkten”. PhD thesis. RWTH Aachen, 2010.
- c32. M. Große-Rhode. “Integrating Semantics for Object-Oriented System Models”. In: *Proc. 28th Intl. Coll. Automata, Languages and Programming (ICALP’01)*. Ed. by F. Orejas, P. G. Spirakis, and J. van Leeuwen. Lect. Notes Comp. Sci. 2076. Springer, 2001, pp. 40–60.
- c33. M. Große-Rhode. *Semantic Integration of Heterogeneous Software Specifications*. Monographs in Theoretical Computer Science. Springer, 2004.

- c34. Y. Hammal. “A Modular State Exploration and Compatibility Checking of UML Dynamic Diagrams”. In: *Proc. 6th ACS/IEEE Intl. Conf. Computer Systems and Applications (AICCSA’08)*. IEEE, 2008, pp. 793–800.
- c35. J. H. Hausmann. “Dynamic Meta Modeling. A Semantics Description Technique for Visual Modeling Languages”. PhD thesis. Universität Paderborn, 2005.
- c36. F. Hilken, P. Niemann, M. Gogolla, and R. Wille. “Filmstripping and Unrolling: A Comparison of Verification Approaches for UML and OCL Behavioral Models”. In: *Proc. 8th Intl. Conf. Tests and Proofs (TAP’14)*. Ed. by M. Seidl and N. Tillmann. Lect. Notes Comp. Sci. 8570. Springer, 2014, pp. 99–116.
- c37. F. Hilken, P. Niemann, M. Gogolla, and R. Wille. “From UML/OCL to Base Models: Transformation Concepts for Generic Validation and Verification”. In: *Proc. 8th Intl. Conf. Theory and Practice of Model Transformations (ICMT’15)*. Ed. by D. S. Kolovos and M. Wimmer. Lect. Notes Comp. Sci. 9152. Springer, 2015, pp. 149–165.
- c38. F. Hilken, P. Niemann, R. Wille, and M. Gogolla. “Towards a Base Model for UML and OCL Verification”. In: *Proc. 11th Ws. Model-Driven Engineering, Verification and Validation (MoDEVVA’14)*. Ed. by F. Boulanger, M. Famelis, and D. Ratiu. CEUR Ws. Proc. 1235. 2014, pp. 59–68.
- c39. Z. Khai, A. Nadeem, and G.-s. Lee. “A Prolog Based Approach to Consistency Checking of UML Class and Sequence Diagrams”. In: *Proc. Intl. Conf.s ASEA, DRBC, EL as part of Future Generation Information Technology Conf. (FGIT’11)*. Ed. by T.-H. Kim, H. Adeli, H.-K. Kim, H.-J. Kang, K. J. Kim, K. Akingbehin, and B. H. Kang. Comm. Comp. Inf. Sci. 257. Springer, 2011, pp. 85–96.
- c40. A. H. Khan and I. Porres. “Consistency of UML Class, Object and Statechart Diagrams Using Ontology Reasoners”. In: *J. Vis. Lang. Comp.* 26 (2015), pp. 42–65.
- c41. S.-K. Kim and D. Carrington. “A Formal Object-oriented Approach to Defining Consistency Constraints for UML Models”. In: *Proc. 15th Austral. Software Engineering Conf. (ASWEC’04)*. IEEE, 2004, pp. 87–94.
- c42. A. Knapp, T. Mossakowski, and M. Roggenbach. “An Institutional Framework for Heterogeneous Formal Development in UML. A Position Paper”. In: *Software, Services, and Systems. Essays Dedicated to Martin Wirsing on the Occasion of His Retirement from the Chair of Programming and Software Engineering*. Ed. by R. D. Nicola and R. Hennicker. Lect. Notes Comp. Sci. 8950. Springer, 2015, pp. 215–230.
- c43. A. Knapp and J. Wuttke. “Model Checking of UML 2.0 Interactions”. In: *Reports Rev. Sel. Papers Ws.s Symp.s MoDELS 2006*. Ed. by T. Kühne. Lect. Notes Comp. Sci. 4364. Springer, 2007, pp. 42–51.
- c44. S. Kuske, M. Gogolla, H.-J. Kreowski, and P. Ziemann. “Towards an Integrated Graph-based Semantics for UML”. In: *Softw. Syst. Model.* 8.3 (2009), pp. 403–422.
- c45. J. M. Küster. “Consistency Management of Object-Oriented Behavioral Models”. PhD thesis. Universität Paderborn, 2004.
- c46. J. M. Küster and G. Engels. “Consistency Management Within Model-Based Object-Oriented Development of Components”. In: *Rev. Lect. 2nd Intl. Symp. Formal Methods for Components and Objects (FMCO’03)*. Ed. by F. S. de Boer, M. M. Bonsangue, S. Graf, and W. P. de Roever. Lect. Notes Comp. Sci. 3188. Springer, 2004, pp. 157–176.
- c47. M. Kyas, H. Fecher, F. S. de Boer, J. Jacob, J. Hooman, M. van der Zwaag, T. Arons, and H. Kugler. “Formalizing UML Models and OCL Constraints in PVS”. In: *Proc. Ws. Semantic Foundations of Engineering Design Languages (SFEDL’04)*. Ed. by G. Lüttgen and M. Mendler. Electr. Notes Theo. Comp. Sci. 115. Elsevier, 2005, pp. 39–47.
- c48. R. Laleau and F. Polack. “Using Formal Metamodels to Check Consistency of Functional Views in Information Systems Specification”. In: *J. Inf. Softw. Techn.* 50.7-8 (2008), pp. 797–814.

- c49. V. S. W. Lam and J. Padget. “Consistency Checking of Sequence Diagrams and Statechart Diagrams Using the π -calculus”. In: *Proc. 5th Intl. Conf. Integrated Formal Methods (IFM’05)*. Ed. by J. Romijn, G. Smith, and J. van de Pol. Lect. Notes Comp. Sci. 3771. Springer, 2005, pp. 347–365.
- c50. X. Li. “A Characterization of UML Diagrams and their Consistency”. In: *Proc. 11th IEEE Intl. Conf. Engineering of Complex Computer Systems (ICECCS’06)*. IEEE. 2006, pp. 67–76.
- c51. B. Litvak, S. S. Tyszberowicz, and A. Yehudai. “Behavioral Consistency Validation of UML Diagrams”. In: *Proc. 1st Intl. Conf. Software Engineering and Formal Methods (SEFM’03)*. IEEE. 2003, pp. 118–125.
- c52. W. Liu, S. Easterbrook, and J. Mylopoulos. “Rule-based Detection of Inconsistency in UML Models”. In: *Proc. Ws. Consistency Problems in UML-based Software Development*. 2002, pp. 106–123.
- c53. Z. Liu, J. He, and X. Li. “Towards a Rigorous Approach to UML-based Development”. In: *Proc. 7th Brazilian Symp. Formal Methods (SBMF’04)*. Ed. by A. Mota and A. Moura. Electr. Notes Theo. Comp. Sci. 130. Elsevier, 2005, pp. 57–77.
- c54. Q. Long, Z. Liu, X. Li, and J. He. “Consistent Code Generation from UML Models”. In: *Proc. 16th Austral. Software Engineering Conf. (ASWEC’05)*. IEEE. 2005, pp. 23–30.
- c55. F. J. Lucas Martínez and J. A. Toval Álvarez. “A Precise Approach for the Analysis of the UML Models Consistency”. In: *Proc. Perspectives in Conceptual Modeling (ER’05) Ws.s.* Ed. by J. Akoka, S. W. Liddle, I.-Y. Song, M. Bertolotto, I. Comyn-Wattiau, S. S.-S. Cherfi, W.-J. van den Heuvel, B. Thalheim, M. Kolp, P. Bresciani, J. Trujillo, C. Kop, and H. C. Mayr. Lect. Notes Comp. Sci. 3770. Springer, 2005, pp. 74–84.
- c56. W. E. McUmber and B. H. Cheng. “A General Framework for Formalizing UML with Formal Languages”. In: *Proc. 23rd Intl. Conf. Software Engineering (ICSE’01)*. IEEE. 2001, pp. 433–442.
- c57. S. J. Mellor and M. J. Balcer. *Executable UML: A Foundation for Model-driven Architecture*. Addison-Wesley, 2002.
- c58. A. Nimiya, T. Yokogawa, H. Miyazaki, S. Amasaki, Y. Sato, and M. Hayase. “Model Checking Consistency of UML Diagrams using Alloy”. In: *WASET Intl. J. Comp., Electr., Autom., Contr., Inf. Eng.* 4.11 (2010), pp. 1696–1699.
- c59. I. Ober and I. Dragomir. “Unambiguous UML Composite Structures: The OMEGA2 Experience”. In: *Proc. 37th Conf. Current Trends in Theory and Practice of Computer Science (SOFSEM’11)*. Ed. by I. Cerná, T. Gyimóthy, J. Hromkovic, K. G. Jeffery, R. Královic, M. Vukolic, and S. Wolf. Lect. Notes Comp. Sci. 6543. Springer, 2011, pp. 418–430.
- c60. Object Management Group. *Semantics of a Foundational Subset for Executable UML Models (fUML)*. Standard formal/2016-01-05. Version 1.2.1. OMG, 2016. URL: <http://www.omg.org/spec/FUML/1.2.1>.
- c61. D. D. Okalas Ossami, J.-P. Jacquot, and J. Souquières. “Consistency in UML and B Multi-view Specifications”. In: *Proc. 5th Intl. Conf. Integrated Formal Methods (IFM’05)*. Ed. by J. Romijn, G. Smith, and J. van de Pol. Lect. Notes Comp. Sci. 3771. Springer, 2005, pp. 386–405.
- c62. G. O’Keefe. “Dynamic Logic Semantics for UML Consistency”. In: *Proc. 2nd Europ. Conf. Model Driven Architecture — Foundations and Applications (ECMDA-FA’06)*. Ed. by A. Rensink and J. Warmer. Lect. Notes Comp. Sci. 4066. Springer, 2006, pp. 113–127.
- c63. P. J. Puczynski. “Checking Consistency between Interaction Diagrams and State Machines in UML Models”. Master thesis. Danmarks Tekniske Universitet, 2012.
- c64. H. Rasch and H. Wehrheim. “Checking Consistency in UML Diagrams: Classes and State Machines”. In: *Proc. 6th IFIP WG 6.1 Intl. Conf. Formal Methods for Open Object-Based Distributed Systems (FMOODS’03)*. Ed. by E. Najm, U. Nestmann, and P. Stevens. Lect. Notes Comp. Sci. 2884. Springer, 2003, pp. 229–243.

- c65. H. Rasch and H. Wehrheim. "Checking the Validity of Scenarios in UML Models". In: *Proc. 7th IFIP WG 6.1 Intl. Conf. Formal Methods for Open Object-Based Distributed Systems (FMOODS'05)*. Ed. by M. Steffen and G. Zavattaro. Lect. Notes Comp. Sci. 3535. Springer, 2005, pp. 67–82.
- c66. G. Reggio, M. Cerioli, and E. Astesiano. "Towards a Rigorous Semantics of UML Supporting Its Multiview Approach". In: *Proc. 4th Intl. Conf. Fundamental Approaches to Software Engineering (FASE'01)*. Ed. by H. Hußmann. Lect. Notes Comp. Sci. 2029. Springer, 2001, pp. 171–186.
- c67. P. G. Sapna and H. Mohanty. "Ensuring Consistency in Relational Repository of UML Models". In: *Proc. 10th Intl. Conf. Information Technology (ICIT'07)*. IEEE. 2007, pp. 217–222.
- c68. Y. Shinkawa. "Inter-model Consistency in UML Based on CPN Formalism". In: *Proc. 13th Asia Pacific Software Engineering Conf. (APSEC'06)*. IEEE. 2006, pp. 411–418.
- c69. C. Soltenborn. "Quality Assurance with Dynamic Meta Modeling". PhD thesis. Universität Paderborn, 2013.
- c70. G. Spanoudakis and H. Kim. "Diagnosis of the Significance of Inconsistencies in Object-oriented Designs: A Framework and Its Experimental Evaluation". In: *J. Syst. Softw.* 64.1 (2002), pp. 3–22.
- c71. R. van der Straeten. "Inconsistency Detection between UML Models Using RACER and nRQL". In: *Proc. 3rd Intl. Ws. Applications of Description Logics (KI'04)*. Ed. by S. Bechhofer, V. Haarslev, C. Lutz, and R. Moeller. CEUR Ws. Proc. 115. 2004.
- c72. R. van der Straeten, J. Simmonds, and T. Mens. "Detecting Inconsistencies between UML Models Using Description Logic". In: *Proc. Intl. Ws. Description Logics (DL'03)*. Ed. by D. Calvanese, G. D. Giacomo, and E. Franconi. CEUR Ws. Proc. 81. 2003.
- c73. R. Wagner, H. Giese, and U. A. Nickel. "A Plug-In for Flexible and Incremental Consistency Management". In: *Proc. 3rd Ws. Consistency Problems in UML-based Software Development*. Ed. by L. Kuzniarz, G. Reggio, J.-L. Sourrouille, Z. Huzar, and M. Staron. Blekinge Inst. Techn. Research Report 2003:06. 2003.
- c74. H. Wang, T. Feng, J. Zhang, and K. Zhang. "Consistency Check between Behaviour Models". In: *Proc. 5th IEEE Intl. Symp. Communications and Information Technology (ISCIT'05)*. IEEE. 2005, pp. 486–489.
- c75. M. Wirsing and A. Knapp. "View Consistency in Software Development". In: *Rev. Papers 9th Intl. Monterey Ws. Radical Innovations of Software and Systems Engineering in the Future*. Ed. by M. Wirsing, A. Knapp, and S. Balsamo. Lect. Notes Comp. Sci. 2941. Springer, 2004, pp. 341–357.
- c76. J. Yang, Q. Long, Z. Liu, and X. Li. "A Predicative Semantic Model for Integrating UML Models". In: *Rev. Sel. Papers 1st Intl. Coll. Theoretical Aspects of Computing (ICTAC'04)*. Ed. by Z. Liu and K. Araki. Lect. Notes Comp. Sci. 3407. Springer, 2005, pp. 170–186.
- c77. S. Yao and S. M. Shatz. "Consistency Checking of UML Dynamic Models Based on Petri Net Techniques". In: *Proc. 15th Intl. Conf. Computing (CIC'06)*. IEEE. 2006.
- c78. W. L. Yeung. "Checking Consistency between UML Class and State Models Based on CSP and B". In: *J. Univ. Comp. Sci.* 10.11 (2004), pp. 1540–1559.
- c79. X. Zhao, Q. Long, and Z. Qiu. "Model Checking Dynamic UML Consistency". In: *Proc. 8th Intl. Conf. Formal Engineering Methods (ICFEM'06)*. Ed. by Z. Liu and J. He. Lect. Notes Comp. Sci. 4260. Springer, 2006, pp. 440–459.

References

1. D. Allaki, M. Dahchour, and A. En-Nouary. "A New Taxonomy of Inconsistencies in UML Models with their Detection Methods for Better MDE". In: *Intl. J. Comp. Sci. Appl.* 12.1 (2015), pp. 48–65.
2. P. Amaya, C. Gonzalez, and J. M. Murillo. "Towards a Subject-Oriented Model-Driven Framework". In: *Proc. 1st Ws. Aspect-Based and Model-Based Separation of Concerns in Software Systems (ABMB'05)*. Ed. by M. Aksit and E. Roubtsova. Electr. Notes Theo. Comp. Sci. 163. 2006, pp. 31–44.
3. R. S. Bashir, S. P. Lee, S. U. R. Khan, S. Farid, and V. Chang. "UML Models Consistency Management: Guidelines for Software Quality Manager". In: *Intl. J. Information Management* 36.6, Part A (2016), pp. 883–899.
4. A. Boronat, A. Knapp, J. Meseguer, and M. Wirsing. "What Is a Multi-modeling Language?" In: *Rev. Sel. Papers 19th Intl. Ws. Recent Trends in Algebraic Development Techniques (WADT'08)*. Ed. by A. Corradini and U. Montanari. Lect. Notes Comp. Sci. 5486. Springer, 2008, pp. 71–87.
5. R. Breu, R. Grosu, F. Huber, B. Rumpe, and W. Schwerin. "Systems, Views and Models of UML". In: *Unified Modeling Language, Technical Aspects and Applications*. Ed. by M. Schader and A. Korthaus. Physica Verlag, 1998.
6. J. Cabot, R. Clarisó, and D. Riera. "On the Verification of UML/OCL Class Diagrams Using Constraint Programming". In: *J. Syst. Softw.* 93 (2014), pp. 1–23.
7. S. Cook, A. Kleppe, R. Mitchell, B. Rumpe, J. Warmer, and A. C. Wills. "Defining UML Family Members Using Prefaces". In: *Proc. 32nd Intl. Conf. Technology of Object-Oriented Languages (TOOLS'99)*. Ed. by C. Mingins and B. Meyer. IEEE, 1999, pp. 102–114.
8. R. M. Dijkman. "Consistency in Multi-Viewpoint Architectural Design". PhD thesis. Universiteit Twente, 2006.
9. J. Dingel, Z. Diskin, and A. Zito. "Understanding and Improving UML Package Merge". In: *Softw. Syst. Model.* 7.4 (2008), pp. 443–467.
10. B. Dobing and J. Parsons. "Dimensions of UML Diagram Use: Practitioner Survey and Research Agenda". In: *Principle Advancements in Database Management Technologies: New Applications and Frameworks*. Ed. by K. Siau and J. Erickson. IGI Publishing, 2010, pp. 271–290.
11. P. van Emde Boas. "Formalizing UML: Mission Impossible?" In: *Proc. OOPSLA'98 Ws. Formalizing UML: Why? How?* Ed. by L. Andrade, A. Moreira, A. Deshpande, and S. Kent. 1998.
12. G. Engels, R. Heckel, G. Taentzer, and H. Ehrig. "A Combined Reference Model- and View-Based Approach to System Specification". In: *Intl. J. Softw. Eng. Knowl. Eng.* 7.4 (1997), pp. 457–477.
13. A. Evans, K. Lano, R. France, and B. Rumpe. "Meta-Modeling Semantics of UML". In: *Behavioral Specifications of Businesses and Systems*. Ed. by H. Kilov, B. Rumpe, and I. Simmonds. Kluwer Academic Publisher, 1999. Chap. 4, pp. 45–60.
14. A. Evans, K. Lano, R. France, and B. Rumpe. "The UML as a Formal Modeling Notation". In: *Sel. Papers 1st Intl. Ws. Unified Modeling Language (UML'98)*. Ed. by J. Bézivin and P.-A. Muller. Lect. Notes Comp. Sci. 1618. Springer, 1999, pp. 336–348.
15. J. A. Goguen and R. M. Burstall. "Institutions: Abstract Model Theory for Specification and Programming". In: *J. ACM* 39 (1992), pp. 95–146.
16. J. A. Goguen and G. Roşu. "Institution Morphisms". In: *Formal Asp. Comp.* 13 (2002), pp. 274–307.

17. R. von Hanxleden, E. A. Lee, C. Motika, and H. Fuhrmann. “Multi-View Modeling and Pragmatics in 2020”. In: *Rev. Sel. Papers 17th Monterey Ws.* Ed. by R. Calinescu and D. Garlan. Lect. Notes Comp. Sci. 7539. Springer, 2012, pp. 209–223.
18. F. Hilken, P. Niemann, M. Gogolla, and R. Wille. “Towards a Catalog of Structural and Behavioral Verification Tasks for UML/OCL Models”. In: *Proc. Modellierung 2016.* Ed. by A. Oberweis and R. H. Reussner. Lect. Notes Inf. 254. GI, 2016, pp. 117–124.
19. V. Hoffmann, H. Lichter, A. Nyßen, and A. Walter. “Towards the Integration of UML and Textual Use Case Modeling”. In: *J. Obj. Techn.* 8.3 (2009), pp. 85–100.
20. Z. Huzar, L. Kuzniarz, G. Reggio, and J.-L. Sourrouille. “Consistency Problems in UML-Based Software Development”. In: *Rev. Sel. Papers UML’04 Satellite Activities.* Ed. by N. J. Nunes, B. Selic, A. R. da Silva, and J. A. T. Álvarez. Lect. Notes Comp. Sci. 3297. Springer, 2005, pp. 1–12.
21. IEEE Standards Association. *Recommended Practice for Architectural Description for Software-Intensive Systems.* Standard 1471-2000. IEEE Computer Society, 2000.
22. International Organization for Standardization. *Systems and Software engineering — Architecture description.* Standard 42010:2011. ISO/IEC/IEEE, 2011.
23. D. Kholkar, G. M. Krishna, U. Shrotri, and R. Venkatesh. “Visual Specification and Analysis of Use Cases”. In: *Proc. ACM Symp. Software Visualization (SOFTVIS’05).* Ed. by T. L. Naps and W. D. Pauw. ACM, 2005, pp. 77–85.
24. S.-K. Kim and D. Carrington. “Formalizing the UML Class Diagram Using Object-Z”. In: *Proc. 2nd Intl. Conf. Unified Modeling Language (UML’99).* Ed. by R. France and B. Rumpe. Lect. Notes Comp. Sci. 1723. Springer, 1999, pp. 83–98.
25. O. Kutz and T. Mossakowski. “A Modular Consistency Proof for Dolce”. In: *Proc. 25th AAAI Conf. Artificial Intelligence and 23rd Innovative Applications of Artificial Intelligence Conf.* Ed. by W. Burgard and D. Roth. AAAI Press, 2011, pp. 227–234.
26. P. Langer, T. Mayerhofer, M. Wimmer, and G. Kappel. “On the Usage of UML: Initial Results of Analyzing Open UML Models”. In: *Proc. Modellierung 2014.* Ed. by H.-G. Fill, D. Karagiannis, and U. Reimer. Lect. Notes Inf. 225. GI, 2014, pp. 289–304.
27. D. Latella, I. Majzik, and M. Massink. “Automatic Verification of a Behavioural Subset of UML Statechart Diagrams Using the SPIN Model-checker”. In: *Formal Asp. Comput.* 11.6 (1999), pp. 637–664.
28. F. J. Lucas, F. Molina, and A. Toval. “A Systematic Review of UML Model Consistency Management”. In: *J. Inf. Softw. Techn.* 51.12 (2009), pp. 1631–1645.
29. T. Mens, R. van der Straeten, and J. Simmonds. “A Framework for Managing Consistency of Evolving UML Models”. In: *Software Evolution with UML and XML.* Ed. by H. Yang. Idea Group, 2005. Chap. 1, pp. 1–30.
30. R. G. Mohammadi and A. A. Barforoush. “Enforcing Component Dependency in UML Deployment Diagram for Cloud Applications”. In: *Proc. 7th Intl. Symp. Telecommunications (IST’14).* IEEE, 2014, pp. 412–417.
31. T. Mossakowski and A. Tarlecki. “Heterogeneous Logical Environments for Distributed Specifications”. In: *Rev. Sel. Papers 19th Intl. Ws. Recent Trends in Algebraic Development Techniques (WADT’08).* Ed. by A. Corradini and U. Montanari. Lect. Notes Comp. Sci. 5486. Springer, 2009, pp. 266–289.
32. F. Munker, A. Albers, D. Wagner, and M. Behrendt. “Multi-View Modeling in SysML: Thematic Structuring for Multiple Thematic Views”. In: *Proc. Conf. Systems Engineering Research (CSER’14).* Ed. by A. M. Madni, B. Boehm, M. Sievers, and M. Wheaton. Procedia Comp. Sci. 28. Elsevier, 2014, pp. 531–538.
33. Object Management Group. *Object Constraint Language.* Standard formal/2014-02-03. Version 2.4. OMG, 2014. URL: <http://www.omg.org/spec/OCL/2.4>.

34. Object Management Group. *Unified Modeling Language*. Standard formal/2015-03-01. Version 2.5. OMG, 2015. URL: <http://www.omg.org/spec/UML/2.5>.
35. R. F. Paige, P. J. Brooke, and J. S. Ostroff. "Metamodel-based Model Conformance and Multiview Consistency". In: *ACM Trans. Softw. Eng. Meth.* 16.3, 11 (2007).
36. Z. Pap, I. Majzik, A. Pataricza, and A. Szegi. "Completeness and Consistency Analysis of UML Statechart Specifications". In: *Proc. IEEE Ws. Design and Diagnostics of Electronic Circuits and Systems (DDECS'01)*. IEEE. 2001, pp. 83–90.
37. Z. Pap, I. Majzik, A. Pataricza, and A. Szegi. "Methods of Checking General Safety Criteria in UML Statechart Specifications". In: *Rel. Eng. & Sys. Safety* 87.1 (2005), pp. 89–107.
38. A. Schürr and A. J. Winter. "Formal Definition and Refinement of UML's Module/Package Concept". In: *ECOOP'97 Ws. Reader*. Ed. by J. Bosch and S. Mitchell. Lect. Notes Comp. Sci. 1357. Springer, 1998, pp. 211–215.
39. A. A. Shah, A. A. Kerzhner, D. Schaefer, and C. J. J. Paredis. "Multi-View Modeling to Support Embedded Systems Engineering in SysML". In: *Graph Transformations and Model-Driven Engineering. Essays Dedicated to Manfred Nagl on the Occasion of his 65th Birthday*. Ed. by G. Engels, C. Lewerentz, W. Schäfer, A. Schürr, and B. Westfechtel. Lect. Notes Comp. Sci. 5765. Springer, 2010, pp. 580–601.
40. D. Torre, Y. Labiche, and M. Genero. "UML Consistency Rules: A Systematic Mapping Study". In: *Proc. 18th Intl. Conf. Evaluation and Assessment in Software Engineering (EA-SE'14)*. ACM. 2014.
41. D. Torre, Y. Labiche, M. Genero, and M. Elaasar. *A Systematic Identification of Consistency Rules for UML Diagrams*. Technical Report SCE-15-01. Carleton University, 2016.
42. M. Usman, A. Nadeem, T.-h. Kim, and E.-s. Cho. "A Survey of Consistency Checking Techniques for UML Models". In: *Proc. Advanced Software Engineering and Its Applications (ASEA'08)*. IEEE. 2008, pp. 57–62.