A Model-Prover for Constrained Dynamic Conversations

Diletta Cacciagrano
University of Camerino
Math. & Comp. Science Dep.
Camerino, Italy
diletta.cacciagrano@unicam.it

Flavio Corradini
University of Camerino
Math. & Comp. Science Dep.
Camerino, Italy
flavio.corradini@unicam.it

Rosario Culmone
University of Camerino
Math. & Comp. Science Dep.
Camerino, Italy
rosario.culmone@unicam.it

Luca Tesei
University of Camerino
Math. & Comp. Science Dep.
Camerino, Italy
luca.tesei@unicam.it

Leonardo Vito
University of Camerino
Math. & Comp. Science Dep.
Camerino, Italy
leonardo.vito@unicam.it

ABSTRACT

In a service-oriented architecture, systems communicate by exchanging messages. In this work, we propose a formal model based on OCL-constrained UML Class diagrams and a methodology based on Alloy Analyzer respectively for describing and verifying any first-order constrained client-server conversations. This framework allows us to verify conversation protocol designs at a fairly detailed level and to check first-order logic constraints on both message flows and message contents.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Client/server; D.2.1 [**Software Engineering**]: Methodologies; D.2.2 [**Software Engineering**]: Design tools and techniques; D.2.4 [**Software Engineering**]: Software/Program verification, Formal methods, Model checking, Validation

Keywords

Web Service, Conversations, WSDL, UML, OCL, Alloy

1. INTRODUCTION

The recent trend in Web Services is fostering a scenario where clients perform run time queries in search of services, services provide some given capabilities, and both systems communicate by exchanging messages. Message passing is a mechanism for robust and loosely coupled interactions which, differently from traditional RPC models, is not based on a fairly rigid request-response interaction style. The set of related messages exchanged by multiple interacting parties is called *conversation*; in particular, a *client-server conversation* is a special case where only two interacting parties are involved. The Web Services Description Language (WSDL) [11] is the standard used for publishing abstract and concrete descriptions of Web Services - including the schemas of exchanged messages, the name and type of operations that the service exposes and some simple interaction patterns. On the other hand, there are a multitude of specifications for describing conversation rules - [2], [9], [8] and [10] are few examples - each of them defining a structured language expressing (temporal, priority, etc.) relationships between the exchanged messages.

Different models have been defined in order to specify and verify the behavior of a service in terms of flow of exchanged messages[1]. For example, in [18] mediated composite services specified in BPEL are verified against the design specified using Message Sequence Chart and Finite State Process notations, while in [16, 19] finite automata are augmented with XML messages, XPath [12] expressions and boolean conditions, in order to verify temporal properties of the conversations of single and composite Web Services[2].

2. FRAMEWORK AND METHODOLOGY

In this scenario, we propose a formal model for describing and verifying any *first-order constrained client-server* conversation. The model is independent from the conversation language: we only assume a generic XML-based document describing conversations and WSDL [11] describing message schemas[3]. As in [17], constraints are CLiX [3] rules (i) abling/disabling message transitions in the conversation flow and (ii) well-typing the involved messages. The verification procedure relies on Alloy [1], an *object-oriented*, *first-*

iiWAS2008, November 24 - 26, 2008, Linz, Austria

[1] In terms of flow of exchanged messages, the behavior of a service describes the changes of its states; message-, activity- and event-based specifications are well-known formal models, relying on different kinds of actions to change state.

[2] The verification framework proposed in [16, 19] is based on SPIN [22] and inputs BPEL specifications of Web Services translated into PROMELA, a boolean-logic based language: for this reasons, it can only achieve partial verifications by fixing the sizes of the input queues in the translation, and complete verifications only under stronger conditions.

[3] The proposed framework also fits on a scenario in which message templates are described by XML schemas [7].

order logic-based modeling language, equipped with an analyzer providing a unique hybrid of features associated with *theorem provers* as well as *model checkers*. At this aim, the fully XML-based model (i.e., conversation, WSDL and CLiX documents) is encoded into an OCL-constrained UML Class Diagram, making possible by Alloy (i) to verify the conversation design at a fairly detailed level, both on message flow and on message contents, and (ii) to check constraint configurations, both generic (consistency constraints) and specific (customized to the conversation). Modeling a conversation as an OCL-constrained UML Class Diagram has an interesting consequence: it is possible to build an incremental verification procedure in Alloy, testing the diagram initially equipped with only one constraint - if there exists - then enriching the previous diagram with one more constraint only after a successful verification result, and so on. It follows that the *global* verification procedure is partitioned in *local* steps, since the successful/unsuccessful result of a phase is associated to a well-known constraint. In the following, we explain in detail the main key assumptions.

Linking conversation document and WSDL: We denote by W_c the generic (XML-based) conversation document, by W_m the WSDL document containing the templates of any exchanged message, and by $\mathcal{G}$ the set of CLiX rules constraining W_c and W_m XML-elements. Then, we state a relationship, called *stability*, between W_c and W_m as follows: for each operation o in W_m, the schema associated to each input, resp. output/fault, operation element p in o is the schema of an inbound, resp. outbound, message type m in W_c. Side effects of this assumption are the following: (i) the scope of any rule in $\mathcal{G}$ only involves W_m operation element schemas, and (ii) there is a syntactic match between message XML-identifiers in W_c and operation element XML-identifiers in W_m, making possible to rightly encode CLiX rule into OCL constraints (and vice versa)[4].

(First-order) Guarded automata and UML Class Diagrams: In [16, 19] it has been proved that any conversation can be modeled as a *boolean* guarded automaton. Our framework is based on an extension of this model, obtained imposing CLiX[5] rules as *first-order logic* guards. A conversation is modeled by a so-called *Constraint Diagram*, i.e. a UML Class Diagram equipped with OCL [4] constraints. Intuitively, a Constraint Diagram is an UML specification of the guarded automaton associated to a conversation: i) each class represents a message type, (ii) two classes m_1 and m_2 are related if there is a state q and two transitions, respectively labeled with m_1 and incoming to q and labeled with m_2 and outgoing to q, and (iii) OCL constraints correspond to CLiX guards.

Substantially, a Constraint Diagram is an UML specification of a conversation *on its own*, i.e. without reasoning in terms of guarded automata: to model a - both existing and novel - conversation by a Constraint Diagram, it suffices to define classes, associations and OCL constraints in such a way that (i) each class models a message type, (ii) associations among classes correspond to interactions involving message types associated to these classes, and (iii) constraints are OCL formulas on class attributes, those classes being associated to messages to constrain. The main reasons of replacing a guarded automata-based model by an OCL-constrained Class Diagram consist of the following points: (i) differently from other UML models, a Class Diagram is suitable for describing and for designing respectively existent and novel conversation protocols; (ii) it is a well-known UML diagram which can be annotated by OCL expressions; (iii) it is suitable to be verified by Alloy; (iv) it looks as a suitable specification where automatically importing - in the form of *templates* - OCL constraints expressing consistency properties, i.e invariant for any conversation; (v) it can express properties which first-order logic, UML without OCL and OCL itself cannot. To better explain the last point, it suffices to consider the *transitive closure* property: it is well-known that it cannot be expressed in first-order logic, and also that both UML and OCL have no transitive closure operator. However, UML equipped with OCL constraints attempts to axiomatize the transitive closure operator. As a consequence, it is possible to express a simple property stating that "any defined message type has to be useful" - i.e. it is used in at least one conversation trace - just introducing in $\mathcal{C}d_{\mathbf{W}}$ an empty class `Start` representing an empty message, for every "initial class" - i.e. associated to an initial message - `I` an association from `Start` to `I` and, for every class `X`, an OCL constraint of the following form:

```
context X
def:  tr_closure:  Set(Message) =
      self.next->union(self.next->collect(e | e.tr_closure))
inv:  self in tr_closure(Start)
```

Formalizing UML and OCL in Alloy: Formalizing UML and OCL for the purpose of analysis and verification is a well-known topic: consider the use of B [24], a formalization of OCL in Isabelle/HOL [15], syntactic analyzers [5], simulators [6], compilers enabling run-time checking of specifications [23], model checkers [20] and integrations with theorem provers [14], the USE tool [25] implementing an interpreter of OCL for run-time checking. In the framework we propose here, the translation of UML into Alloy is *fully automatic*[6] thanks to UML2Alloy [13], a filter tool formatting UML Class Diagrams enriched with OCL constraints as Alloy specifications. The current version of UML2Alloy performs the translation creating a text file with the Alloy model; the designer, which knows UML and OCL but maybe does not have any notion about Alloy language syntax, only needs to use the Alloy Analyzer to open the text file and perform the analysis.

3. A MODEL FOR VALID FIRST-ORDER CONSTRAINED CONVERSATIONS

In this section, we formally define a model for *valid* first-order constrained client-server conversations, where *valid* is intended w.r.t. a set of CLiX rules.

Notation 1. We denote by W_c the generic (XML-based) document describing a client-server conversation; by W_m the generic XML-based document containing the templates of any W_c conversation message, and by $\mathcal{G}$ the set of CLiX rules constraining W_c and W_m (message) XML elements; by $M = \{m_k | k \in [1...n], n \geq 1\}$ the finite set of message types involved in W_c and described in W_m; by M_i and M_o the

[4]It is well-known that both OCL and CLiX support first-order logic, and that OCL can be encoded into CLiX.

[5]CLiX is a logical language, used both to constrain XML documents internally and to execute inter-document checks. It allows constraints to be described using a mixture of first-order logic and XPath expressions.

[6]Differently from [21], where the translation is *manual*.

finite sets of respectively inbound and outbound message types in $M = M_i \cup M_o$; by $x(d)$ the XML schema describing an element d.

First, we abstract from the tuple $\langle W_c, W_m, \mathcal{G}\rangle$, replacing it with its guarded automaton-based representation.

Definition 1. The *First-Order guarded (FOG) automaton* associated to $\langle W_c, W_m, \mathcal{G}\rangle$ is the tuple $\mathcal{A}$=$\langle S, M, V, s, f, \delta, \mathcal{G}\rangle$, where:
i. S is a finite set of states;
ii. $M = M_i \cup M_o$ is as above described;
iii. $V = \langle v_1, ..., v_{|M|}\rangle$ is a vector of XML local variables, where $\forall j \in [1...|M|], v_j$ is associated to $m_j \in M$;
iv. $s \in S$ is the initial state and $f \in S$ is the final state;
v. $\mathcal{G} = \{g(q) = g(m_k, \langle d_1, ..., d_{|M|}\rangle)$ CLiX rule$\}$, such that $q \in S$, $m_k \in M$ and $\forall j \in [1...|M|]$, $d_j \in \{d(v_j(q)), \lambda\}$.
vi. $\delta = \{(q, (l, g(q)), Q)\}$ is a *state transition* relation, where $q \in S$, $Q \subseteq S$, $l \in \{m_i |\ m_i \in M_i\} \cup \{\overline{m}_o |\ m_o \in M_o\}$ and $g(q) \in \mathcal{G}$.

Message types and local variables are XML documents. Each local variable v_j in V corresponds to a message types m_j in M. $\forall q \in S$ and $\forall j \in [1...|M|]$, $d(v_j(q))$ denotes the XML document obtained enqueuing all the sent/received (until the state q) message instances that match to the type m_j. Each transition $\tau \in \delta$ is in one of the following two forms:
(*receive-transition*) $\tau = (q_1, (m_k, g(q_1)), Q)$, where $m_k \in M_i$: the transition nondeterministically changes the state of the automaton from q_1 to $q_2 \in Q$, it removes the received message instance (of type m_k) from the input queue and it updates v_k in V, corresponding to m_k, by the concatenation of the received instance, in the case $g(q_1)$ holds;
(*send-transition*) $\tau = (q_1, (\overline{m}_k, g(q_1)), Q)$, where $m_k \in M_o$: the transition nondeterministically changes the state of the automaton from q_1 to $q_2 \in Q$, it appends the sent message instance (of type m_k) to the input queue of the client and it updates $v_k \in V$, corresponding to m_k, by the concatenation of the sent instance, in the case $g(q_1)$ holds.

Definition 2. Let $\mathcal{A} = \langle S, M, V, s, f, \delta, \mathcal{G}\rangle$ be the FOG automaton associated to $\langle W_c, W_m, \mathcal{G}\rangle$. Given a guard $g(q) = g(m_k, \langle d_1, ..., d_{|M|}\rangle) \in \mathcal{G}$, then:
i. $\langle d_1, ..., d_{|M|}\rangle$ denotes the *actual context* of $g(q)$, obtained filtering out all the local variables such that no XML attribute of theirs is involved in $g(q)$.
ii. $X(g(q))$ denotes the *formal context* of $g(q)$, obtained by an XML-schema concatenation of those local variable included in the actual context of $g(q)$, i.e.
$X(g(q)) = \bigodot_{j\in[1...|M|]} x(m_j)$, where $x(m_j) = \lambda$ if $d_j = \lambda$.

Notation 2. Let W_m be a WSDL document. We denote by O_m the set of operation in W_m; for every $o \in O_m$, by $p_{in}(o)$ and $p_{out}(o)$ respectively the input and the output/fault operation elements of o.

We also assume that W_c and W_m are related as follows: for each operation $o \in O_m$, for every $p_k \in p_{in}(o)$ (resp. $p_{out}(o)$), for every $m_k \in M_i$ (resp. M_o), $x(m_k) = x(p_k)$. We formally define this kind of relationship between W_c and W_m as follows.

Definition 3. Let $\mathcal{A} = \langle S, M, V, s, f, \delta, \mathcal{G}\rangle$ be the FOG automaton associated to $\langle W_c, W_m, \mathcal{G}\rangle$, and let W_m be a WSDL document. $\mathbf{W} = \langle \mathcal{A}, W_m\rangle$ is *stable* if and only if $\forall q_1 \in S$ such that $(q_1, (m_{k_1}, g_1(q_1)), Q_1) \in \delta$:
i. $\exists o \in \mathcal{O}_m$ such that $p_{in}(o) = \{p_{k_1}\}$ and $x(p_{k_1}) = x(m_{k_1})$;
ii.$\exists q_2 \in Q_1$, $\exists h\ (2 \le h \le 3)$ s.t. $(q_2, (\overline{m}_{k_h}, g_h(q_2)), Q_h) \in \delta$ iff $p_{out}(o) = \{m_{k_h} |\ 2 \le h \le 3\}$ and $x(p_{k_h}) = x(m_{k_h})$.

The *stability* assumption (Definition 3) implies that it is possible to use everywhere the identifier m_k of message type in place of the identifier p_k of operation element, and that both formal and actual contexts of any guard in $\mathcal{G}$ only involve W_m operation element schemas.

Given $\mathbf{W} = \langle \mathcal{A}, W_m\rangle$ stable, we can built a Class Diagram equipped by OCL constraints, called *Constraint Diagram* and denoted by $\mathcal{C}d_{\mathbf{W}}$, *semantically equivalent* to $\mathbf{W}$, where (i) each class *corresponds* to a message type, (ii) class associations *correspond* to state transitions, and (iii) OCL constraints in $\mathcal{C}d_{\mathbf{W}}$ *correspond* to CLiX guards in $\mathcal{G}$. The *correspondence* between $\mathbf{W} = \langle \mathcal{A}, W_m\rangle$ stable and $\mathcal{C}d_{\mathbf{W}}$ is formally defined as follows.

Definition 4. Given $\mathbf{W} = \langle \mathcal{A}, W_m\rangle$ stable, the *Constraint Diagram* $\mathcal{C}d_{\mathbf{W}}$ associated to $\mathbf{W}$ is a Class Diagram obtained translating $\mathbf{W}$ by the encoding $[\![\cdot]\!]$ so defined:
i. $\forall o \in \mathcal{O}_m$ such that $o \cong$
`<operation name=''O''>`
`<input name=''m_k1'' message=''tns:m_k1-Document''/>`
[*<output name="m_k2" message="tns:m_k2-Document"/>*]
[*<fault name="m_k3" message="tns:m_k3-Document"/>*]
`</operation>`
$\forall m_{k_y} \in (p_{in}(o) \cup p_{out}(o))$ $(1 \le y \le 3)$ such that $m_{k_y} \cong$
`<message name=''m_ky''>`
`<element=''xsd-m_ky:m_ky-Element''/>`
`</message>`
and such that $\exists t_{k_y}$ such that $t_{k_y} \cong$
`<xsd-m_ky:element name=''m_ky-Element''>`
`...`
`</xsd:element>`
then $[\![m_{k_y}]\!] =$`c(m_ky)`, where `c(m_ky)` is the class of t_{k_y}.
ii. $\forall o \in \mathcal{O}_m$:$p_{in}(o)$=$\{m_{k_1}\}$ and $p_{out}(o) = \{m_{k_h} | 2 \le h \le 3\}$, there exists an association between `c(m_1)` and `c(m_kh)`;
iii. $\forall o_1, o_2 \in \mathcal{O}_m$: $p_{in}(o_1)$=$\{m_{k_1}\}$, $p_{out}(o_1)$=$\emptyset$ and $p_{in}(o_2) = \{m_{k_2}\}$, there exists an association from `c(m_k1)` to `c(m_k2)` if and only if $\exists q_1, q_2 \in S$ such that $(q_1, (m_{k_1}, g_1(q_1)), Q_1) \in \delta$, $(q_2, (m_{k_2}, g_2(q_2)), Q_2) \in \delta$ and $q_2 \in Q_1$;
iv. $\forall o_1, o_2 \in \mathcal{O}_m$: $p_{out}(o_1) = \{m_{k_h} | 2 \le h \le 3\}$ and $p_{in}(o_2) = \{m_{k_1}\}$, there exists an association from `c(m_kh)` to `c(m_k1)` iff $\exists q_h \in S$ s.t. $(q_h, (\overline{m}_{k_h}, g_h(q_h)), Q_h) \in \delta$ $(2 \le h \le 3)$ and $\exists q_1 \in Q_h$ s.t. $(q_1, (m_{k_1}, g_1(q_1)), Q_1) \in \delta$.
v. For every guard $g(q)$=$g(m_k, \langle d_1, ..., d_{|M|}\rangle) \in \mathcal{G}$, then
$[\![g(q)]\!] =$ `context` $\bigcup_{m_j} [\![m_j]\!]$ `inv:g`, where $x(m_j)$ in $X(g(q))$ and `g` is a formula semantically equivalent to $g(q)$.

Notice that it would be also possible to start with the design of a novel conversation in the form of Constraint Diagram, building from it a *stable* pair of XML documents.

Example 1. Suppose to project a toy authentication service defining the following scenario: (i) the client is required either to register by a *Registration* form, or to login by a *Login* form; (ii) after filling a *Registration* form, the client can only access to a *Login* one; (iii) after filling a *Login* form, the client is allowed to enter the system only if either it has already registered *in a past session* and login username is

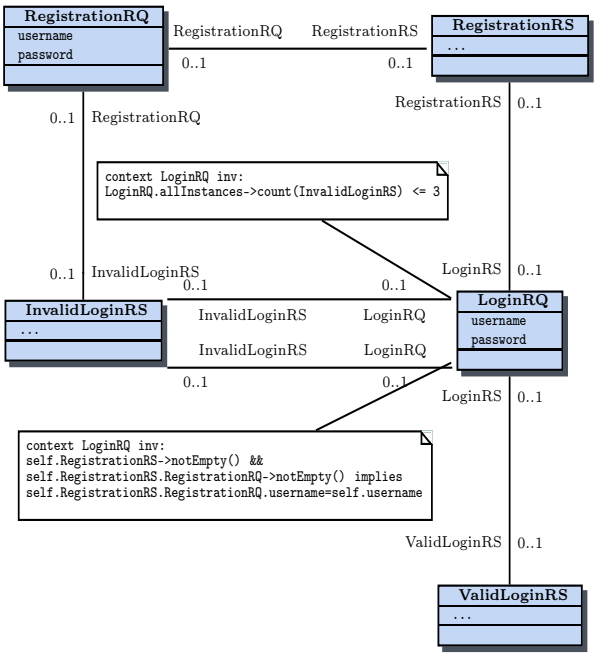

Figure 1: Constraint Diagram.

valid, or he has just filled a *Registration* form *in the current session* and login username is valid; (iv) the allowed max number of failed logins is 3. In terms of WSDL document, we could define a `Login` operation, including `LoginRQ` as inbound element, `ValidLoginRS` and `InvalidLoginRS` as outbound elements, and a `Registration` operation, including `RegistrationRQ` and `RegistrationRS` respectively as inbound and outbound elements. Fig.1 shows the Constraint Diagram $\mathcal{C}d_{\mathbf{W}}$, associated to the protocol above described, which has to be input into UML2Alloy. The attributes of a class correspond to the WSDL attributes of the message described by the class itself. `LoginRQ.allInstances` denotes the set of `LoginRQ` instances, and `LoginRQ.allInstances->count(InvalidLoginRS)` denotes the number of `LoginRQ` instances associated with `InvalidLoginRS` ones.

4. REFERENCES

[1] The Alloy analyzer, http://alloy.mit.edu/.

[2] BPML: Business Process Modeling Language, http://www.bpmi.org/.

[3] CLiX: Constraint Language in XML, http://www.clixml.org/clix/1.0/.

[4] Object Constraint Language Specification V. 2.0, http://www.klasse.nl/ocl/ocl-subm.html/.

[5] OCL 1.4 syntax checker, http://www.klasse.nl/ocl/.

[6] OCLE 1.0, http://lci.cs.ubbcluj.ro/ocle/.

[7] W3C XML Schema, http://www.w3.org/xml/schema/.

[8] WS-CDL: Web Service Choreography Description Language, http://www.w3.org/tr/ws-cdl-10/.

[9] WSCI: Web Service Choreography Interface, http://www.w3.org/tr/wsci/.

[10] WSCL: Web Service Conversation Language, http://www.w3.org/tr/wscl10/.

[11] WSDL: Web Service Definition Language, http://www.w3.org/tr/wsdl/.

[12] XML Path Language (XPath) V. 2.0, http://www.w3.org/tr/xpath20/.

[13] K. Anastasakis, B. Bordbar, G. Georg, and I. Ray. UML2Alloy: A challenging model transformation. pages 436–450. 2007.

[14] D. B. Aredo. A framework for semantics of UML Sequence Diagrams in PVS. *Journal of Universal Computer Science*, 8(7):674–697, 2002.

[15] A. D. Brucker and B. Wolff. A proposal for a formal OCL semantics in Isabelle/HOL. In *TPHOLs '02: Proceedings of the 15th International Conference on Theorem Proving in Higher Order Logics*, pages 99–114, London, UK, 2002. Springer-Verlag.

[16] T. Bultan, X. Fu, R. Hull, and J. Su. Conversation specification: a new approach to design and analysis of e-service composition. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 403–410, New York, NY, USA, 2003. ACM Press.

[17] D. Cacciagrano, F. Corradini, R. Culmone, and L. Vito. Dynamic constraint-based invocation of Web Services. In M. Bravetti, M. Nez, and G. Zavattaro, editors, *WS-FM*, volume 4184 of *Lecture Notes in Computer Science*, pages 138–147. Springer, 2006.

[18] H. Foster, S. Uchitel, J. Magee, and J. Kramer. Model-based verification of Web Service compositions. pages 152–163. IEEE Computer Society, 2003.

[19] X. Fu, T. Bultan, and J. Su. Analysis of interacting BPEL Web Services. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 621–630, New York, NY, USA, 2004. ACM.

[20] M. D. M. Gallardo, P. Merino, and E. Pimentel. Debugging UML designs with model checking. *Journal of Object Technology*, 1:101–117, 2002.

[21] G. Georg, J. Bieman, and R. France. Using Alloy and UML/OCL to specify runtime configuration management: A case study. In *Practical UML-Based Rigorous Development Methods - Countering or Integrating the eXtremists. Volume P-7 of LNI., German Informatics Society*, pages 128–141, 2001.

[22] G. J. Holzmann. The model checker SPIN. *Software Engineering*, 23(5):279–295, 1997.

[23] H. Hussmann, B. Demuth, and F. Finger. Modular architecture for a toolset supporting OCL. In A. Evans, S. Kent, and B. Selic, editors, *UML 2000 - The Unified Modeling Language. Advancing the Standard. Third International Conference, York, UK, October 2000, Proceedings*, volume 1939 of *LNCS*, pages 278–293. Springer, 2000.

[24] R. Marcano and N. Levy. Transformation rules of OCL constraints into B formal expressions. In J. Jürjens, M. V. Cengarle, E. B. Fernandez, B. Rumpe, and R. Sandner, editors, *Critical Systems Development with UML – Proceedings of the UML'02 workshop*, pages 155–162. Technische Universität München, Institut für Informatik, 2002.

[25] M. Richters and M. Gogolla. Validating UML models and OCL constraints. pages 265–277. Springer, 2000.