

Автоматический контроль целостности визуальных спецификаций проекта

Ольхович Л. Б.* Кознов Д. В.†

9 июня 2003 г.

{leo,dim}@tepkom.ru

Аннотация

Данная работа посвящена визуальным языкам проектов — диалектам стандартных языков визуального моделирования (UML, SDL и т.д.), создаваемым с учетом особенностей конкретного проекта по разработке программного обеспечения. Рассматриваются следующие случаи использования визуального моделирования: (а) в качестве средства коммуникации участников процесса разработки, (б) при использовании совместно с автоматическими средствами обработки визуальных спецификаций (генерацией кода, тестов и т.д.). В работе приводятся фрагменты визуальных языков проектов, созданных в рамках различных промышленных разработок. Также предлагается подход для спецификации визуального языка проекта с помощью OCL-ограничений на метамодель UML и автоматической генерации валидаторов, которые можно использовать как в интерактивном, так и в пакетном режиме для контроля соответствия визуальной модели правилам визуального языка проекта.

1 Введение

На данный момент не существует универсального процесса разработки ПО: создано большое количество различных моделей процессов, методов и методик, а также программных средств и линеек программных продуктов. Все эти разнообразные инструменты подразумевают, тем не менее, существенную работу по внедрению в конкретный процесс разработки.

* Работа поддержана грантом Министерства Образования России по исследованиям в области гуманитарных, естественных, технических и медицинских наук.

† Работа поддержана грантом PD02-2.8-145 Министерства Образования России.

В [1] для организации работ по внедрению стандартных средств визуального моделирования (языков, методов, CASE-средств) была предложена методология *CASE-пакета*. Целью ее применения в программном проекте является создание технологического решения — набора стандартных средств визуального моделирования, настроенных на решение проблем данного проекта, а также специально созданных программных средств, дополняющих функциональность стандартных.

В данной работе визуальный язык проекта рассматривается как часть технологического решения, то есть как результат адаптации стандартного визуального языка (UML¹ [2], SDL², IDEF0³, IDEF1x⁴ и т.д.) или группы таких языков к особенностям данного проекта. Этот феномен нуждается в специальном изучении, так как на практике затраты на создание таких языков весьма значительны, а их игнорирование порождает проблемы при практическом использовании визуального моделирования. Тем не менее данный вопрос пока слабо изучен. Существует, например, концепция UML-профилей [2], но она предназначена для адаптации UML к различным типам программного обеспечения — информационным системам, систем реального времени и т.д. — и, фактически, является дальнейшей стандартизацией UML. Идея же визуального языка проекта заключается в создании инструментария, позволяющего конструировать нестандартные диалекты, оптимально учитывающие особенности того или иного проекта. Подобные идеи высказывались ранее, но они либо акцентировались только на способах формального задания диалектов [?], либо носили лишь концептуальный характер [?]. В конечном счете эти идеи не привились в контексте использования современных визуальных языков — UML и SDL, что существенно затрудняет его использование, в частности, в рамках автоматической кодогенерации и других способов на основе автоматизированных процедур.

В работе рассматриваются два вида визуальных языков проекта — для предварительного анализа и проектирования, а также при использовании средств автоматической обработки визуальных спецификаций (генерации конечного кода и тестов, поиск “мертвых” циклов в алгоритмах, оценку непротиворечивости моделей и т.д.).

Предлагается подход к спецификации визуального языка проекта, созданного на основе UML, с помощью “well-formedness rules” — OCL⁵-ограничений

¹Unified Modeling Language — стандарт международной организации OMG (Object Management Group), предназначенный для объектно-ориентированного визуального моделирования ПО.

²Specification and Description Language — стандарт международной организации ITU (International Telecommunication Union), предназначенный для визуального моделирования телекоммуникационных систем.

³Integration Definition For Function Modeling — графический язык и методика его использования, созданные для функционального моделирования бизнес-процессов. Является федеральным стандартом США.

⁴Integration Definition For Information Modeling — графический язык и методика его использования, созданные для моделирования данных для информационных систем. Является федеральным стандартом США.

⁵Object Constraint Language.

на метамодель UML. На основе этих спецификаций предлагается автоматически генерировать *валидаторы* — программные модули, позволяющие проверять соответствие моделей визуальному языку проекта. В работе приводится общая архитектура генератора валидаторов, а также типовая архитектура проектно-ориентированного валидатора.

Данный подход может быть интегрирован с произвольным CASE-пакетом, поддерживающим импорт/экспорт XMI⁶-представления UML-модели, либо ActiveX⁷-интерфейс доступа к ней. Предполагается использовать сгенерированные валидаторы как для интерактивной, так и для пакетной работы — при интеграции в процесс регулярной сборки и регрессионного тестирования.

2 Визуальный язык проекта

Семантика языков визуального моделирования определяется не так строго, как семантика языков программирования. Созданные на языках программирования спецификации предназначены, главным образом, для выполнения вычислителем, а визуальные модели нужны для упрощения человеческих коммуникаций [3]. Каждый конкретный проект обладает своими коммуникативными особенностями, что приводит, в частности, к созданию специальных диалектов при использовании стандартных визуальных языков. Такие диалекты мы будем называть *визуальными языками проектов*.

При использовании языков программирования также создаются диалекты [4], однако, как правило, все их проектно-зависимые изменения совместимы со стандартными средствами автоматической обработки таких языков — компиляторами, отладчиками и т.д. Таким образом, требования операционной корректности являются серьезными ограничениями при создании проектно-ориентированных диалектов языков программирования.

У визуальных языков операционная семантика либо отсутствует (как у UML), либо не является строго обязательной для применения, как, например, в случае SDL⁸. Визуальные языки используются для моделирования отдельных аспектов системы, выбор которых, так же как и определение уровня детализации, производятся исходя из текущих потребностей. Столь же произвольно выбирается для использования определенное подмножество стандартного визуального языка.

Обычно, в случае использования визуального моделирования вне контекста автоматической обработки визуальных спецификаций, визуальный язык проекта определяется только созданными с его помощью спецификациями. То есть, правила использования стандартного визуального языка не

⁶XML Metadata Interchange — стандарт OMG для обмена метаданными, например, для обмена UML-моделями.

⁷Технология Microsoft для взаимодействия приложений.

⁸Многие SDL-спецификации не предназначены для автоматической обработки, являясь, например, составными частями телекоммуникационных стандартов.

определяются отдельно⁹. Однако, если в проекте используется автоматическая обработка визуальных моделей, то особенности созданного диалекта нуждаются в отдельной спецификации вместе с уточнением операционной семантики стандартных конструкций.

Необходимость создания дополнительных проектно-ориентированных средств автоматической обработки визуальных языков проекта контрастирует с практикой использования языков программирования, где обычно достаточно стандартных программных средств. Дело в том, что между визуальными спецификациями и кодом вычислителя существует большой семантический разрыв. Наглядность визуальных спецификаций часто противоречит семантической полноте, необходимой для полноценной генерации. Универсальные решения существуют лишь для отдельных частных случаев — например, при разработке структуры базы данных широко используется пакет ERwin [6] и нотация IDEF1x. Использование стандартных кодогенераторов в большинстве случаев происходит лишь при разовой генерации прототипа системы, который потом дорабатывают “вручную”. Для создания полноценных генерационных решений требуются специальные усилия по адаптации стандартных, либо по созданию специальных средств, позволяющих учитывать специфику проекта и тем самым сглаживающих противоречие между наглядностью визуальных спецификаций и генерацией эффективного кода.

Таким образом, при построении проектно-ориентированных технологических решений на основе визуального моделирования с использованием автоматической обработки визуальных спецификаций к последним выдвигаются жёсткие требования по точности, полноте и непротиворечивости.

3 Примеры визуальных языков проекта

3.1 Пример 1

При анализе и проектировании информационной системы UML использовался следующим образом. С помощью диаграмм случаев использования строились:

- Контекстная модель всей системы.
- Контекстная модель программной части системы.
- Модель функций системы.

С помощью activity-диаграмм была построена модель бизнес-процессов новой системы.

С помощью модели классов была построена:

⁹Подобным образом, единственной спецификацией многих древних языков является набор дошедших до нас текстов, написанных на этих языках [5].

- Концептуальная модель предметной области — связное и доступное изложение основных понятий предметной области, имеющих принципиальное значение для построения информационной системы и в силу этого требующих детального согласования с заказчиком/пользователями системы. Эта модель представляла собой небольшой набор диаграмм, снабженных подробным текстовым описанием.
- Логическая модель предметной области — полная модель предметной области, необходимая для создания базы данных, а также требующая согласования со специалистами предметной области на предмет полноты описания. Не содержит типов атрибутов, справочников, нераскрыты отношениями вида "многие ко многим", многие атрибуты объединены в группы. Эта модель, в отличие от предыдущей, не предназначалась для широкого обсуждения со специалистами предметной области, так как была достаточно сложна для понимания непрограммистами.
- Физическая модель базы данных системы — полная спецификация схемы базы данных с учетом специфики целевой СУБД.

3.2 Пример 2

Технологическое решение RTST [7] было создано для решения ряда проблем большого телекоммуникационного проекта, среди которых была задача по наладке взаимодействия инженеров по оборудованию и программистов. В рамках проекта требовалось реализовать большое количество алгоритмов по управлению аппаратурой, спецификации которых было необходимо согласовывать с инженерами. Спецификации алгоритмов на языках программирования были трудны для понимания и обсуждения, поэтому было принято решение использовать визуальный язык SDL. Кроме того, было решено использовать автоматизированные генераторы для создания исходных кодов системы по SDL-моделям. При этом язык SDL был модифицирован следующим образом:

- Использовалось только подмножество SDL, предназначенное для спецификации поведения. Средства структурной декомпозиции SDL не применялись, все данные описывались на специальном текстовом языке.
- Код в SDL-символах создавался на целевом языке программирования.
- Не использовались SDL-процедуры, т.к. была необходима экономия памяти при реализации SDL-процессов (SDL-процедуры требуют хранения нефиксированного по длине стека вызовов).
- В качестве средств синхронизации процессов использовались только сигналы (то есть, не использовались удаленные процедуры, export- и view- переменные).

Был создан собственный генератор кода, который транслировал модельную спецификацию в целевую платформу с учетом ее специфических ограничений — небольшого размера сегмента кода, необходимость оптимизировать межсегментные вызовы, генерировать различного типа загрузочные модули и т.д.

3.3 Пример 3

Технологическое решение REAL-IT [8] было создано для автоматизированной разработки информационных систем на основе моделирования предметной области с автоматизированной генерацией базы данных и пользовательского интерфейса. При этом использовался язык UML и CASE-пакет REAL [9]. Ниже перечислены основные модификации UML, выполненные в рамках REAL-IT:

- Модель классов была расширена конструкцией, аналогичной конструкции “view” в языке SQL.
- В модели классов в качестве сущностей использовались только классы и view-конструкции.
- Не допускалось наследование от неабстрактных классов.
- Не использовались связи вида “многие ко многим”.
- Пакеты, содержащие долгоживущие (persistent) классы, не могли содержать иных классов.
- Для спецификации ограничений на связи между таблицами использовались диаграммы коопераций.
- Никакие другие диаграммы UML, кроме диаграмм классов и коопераций, не использовались.

4 Обзор подходов к проблеме целостности визуальных моделей

Существует много подходов, где решение проблемы целостности визуальных спецификаций рассматривается как согласованность различных типов моделей — целостность моделей, созданных на основе различных UML-view, согласованное использование UML и SDL, SDL и MSC¹⁰ и т.д. В [10] классифицируются возможные виды несоответствий, возникающих при внесении несогласованных изменений в диаграммы UML. В работах [11], [12] предлагаются способы проверки соответствия между statechart-диаграммами и

¹⁰Message Sequence Chart — стандарт комитета ITU для спецификации сценариев телекоммуникационных систем.

диаграммами последовательностей при помощи систем автоматического доказательства. В работах [13], [14] предлагается способ обнаружения противоречий между диаграммами последовательностей и классов. Эти подходы позволяют повысить согласованность диаграмм, но не учитывают особенностей используемого в каждом конкретном проекте визуального языка.

В проекте REAL [9] соответствие UML-statecharts и диаграмм расширенного конечного автомата SDL было реализовано на основе модификации обоих формализмов и создания общей метамодели [1]. Однако, такой подход привел к серьезным модификациям как UML, так и SDL, сузив выразительные средства обоих языков.

В 1994 году в рамках подхода Syntropy [15] было предложено текстовый язык для уточнения визуальных спецификаций. Однако, этот язык, основанный на математических абстракциях, был сложен для использования. Корпорация IBM адаптировала подход Syntropy и создала на его основе язык OCL, который был включён в стандарт UML 1.1 в 1997 году [16].

Поддержка языка OCL была реализована для ряда CASE-пакетов: например, интерпретатор OCL [17] был встроен в CASE-пакет ArgoUML [18], в рамках проекта KeY [19] реализована поддержка OCL для CASE-средства TogetherCC. Однако, в подобных подходах OCL предназначен для ограничения конкретных UML-моделей.

Начиная с версии 1.4, в стандарте UML вводится понятие профиля (UML Profile). Профиль UML — это адаптация стандартного визуального языка UML к особенностям определённой предметной области [2]. Профили создаются с помощью механизма расширения UML, в том числе, с использованием OCL. В UML 1.4 включены следующие профили — Unified Process и Business Modeling, существуют также и другие, не вошедшие в стандарт.

Использование профилей не заменяет создания визуальных языков проектов. В каждом конкретном случае, используя стандартные средства визуального моделирования и балансируя между наглядностью и корректностью спецификаций, приходится вводить новые конструкции, изменять семантику старых, вводить дополнительные ограничения на использование различных типов диаграмм. Кроме того, профилям крайне недостает согласованной, стандартной поддержки со стороны производителей CASE-пакетов. Так, real-time профиль был реализован компанией Rational в виде отдельного продукта (Rose RealTime), без сохранения преемственности с традиционным UML. Продукт Objecteering UML Profile Builder компании Objecteering Software [23] широко поддерживает работу с профилями, в частности, позволяет создавать новые. Однако вместо OCL для задания ограничений на метамодель UML предлагается использовать нестандартный язык J. Таким образом, использование профилей является пока, скорее, предметом экспериментов и частных решений, реализуемых в рамках отдельных сред визуального моделирования.

В [24] представлена реализация ограничений на метамодель UML на основе OCL. Однако, во-первых, реализация ориентирована на единственный CASE-пакет (ROCASE [25]), во-вторых, она не предназначена для пакетной работы.

Предлагаемый в работе подход отличается от приведенных выше следующим:

- в отличие от профилей UML, ориентирующихся на стандартизацию применения визуального моделирования в рамках отдельных областей, предлагается поддержка индивидуальных решений по использованию визуального моделирования в рамках отдельных проектов.
- предлагается с помощью OCL задавать общие правила использования UML в проекте, а не накладывать ограничения на отдельные фрагменты спецификации;
- предлагается автоматизировать процесс создания валидаторов;
- подход предполагает независимость от конкретного CASE-пакета;
- валидатор предназначается для использования как в интерактивном, так и в пакетном режиме;
- при пакетной работе валидатор позволяет генерировать отчет в форме, удобной как для чтения человеком, так и для автоматической обработки.

5 Использование OCL для спецификации визуального языка проекта

К сожалению, OCL изначально не был ориентирован на автоматическую обработку, и поэтому имеет ряд недостатков, наиболее существенными из которых являются недостаточно точная семантика некоторых его конструкций (так, например, стандартное определение операции *collect* противоречиво), некоторая противоречивость синтаксиса (в частности, невозможно отличить обращение к атрибуту от обращения к одноимённому методу, не принимающему параметров) [26]. Также необходимо упомянуть невозможность полной интеграции OCL в метамодель UML из-за отсутствия метамодели у самого OCL. Некоторые из перечисленных недостатков будут исправлены в OCL2, который войдёт в UML 2.0.

Перечисленные недостатки, однако, не умаляют достоинств OCL, к которым относятся, в первую очередь, простота для понимания и лёгкость в использовании.

В данной работе язык OCL предлагается для использования в качестве средства спецификации визуального языка проекта. Следует отметить, что с его помощью не добавляются новые конструкции — для этого используются стереотипы и прикреплённые значения (*tagged values*) — а лишь накладываются ограничения на существующие.

Ниже приводится ряд фрагментов OCL-спецификаций для визуального языка проекта, созданного в рамках технологического решения REAL-IT [8].

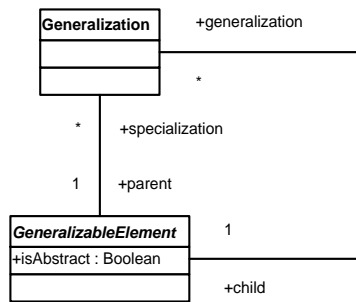


Рис. 1: Фрагмент метамодели 1.

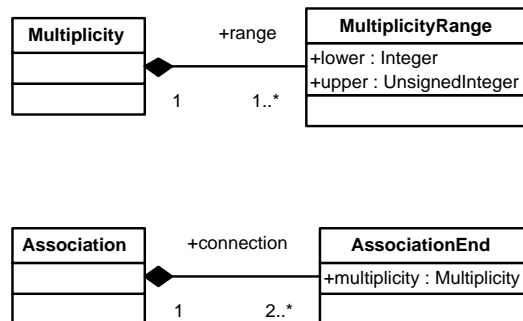


Рис. 2: Фрагмент метамодели 2.

5.1 Пример 1

Данное ограничение заключается в запрете использования наследования для создания потомков неабстрактных классов, что связано с особенностью генерации баз данных в REAL-IT. На рис. 1 представлен фрагмент метамодели UML, на который распространяется это ограничение.

На языке OCL оно специфицируется так:

```

context Generalization
inv: parent.isAbstract
  
```

5.2 Пример 2

Данное ограничение заключается в запрете использования связей вида “многие ко многим”. Оно так же связано с особенностями генерации баз данных в REAL-IT. На рис. 2 представлен соответствующий фрагмент метамодели UML.

Спецификация на языке OCL приведена ниже:

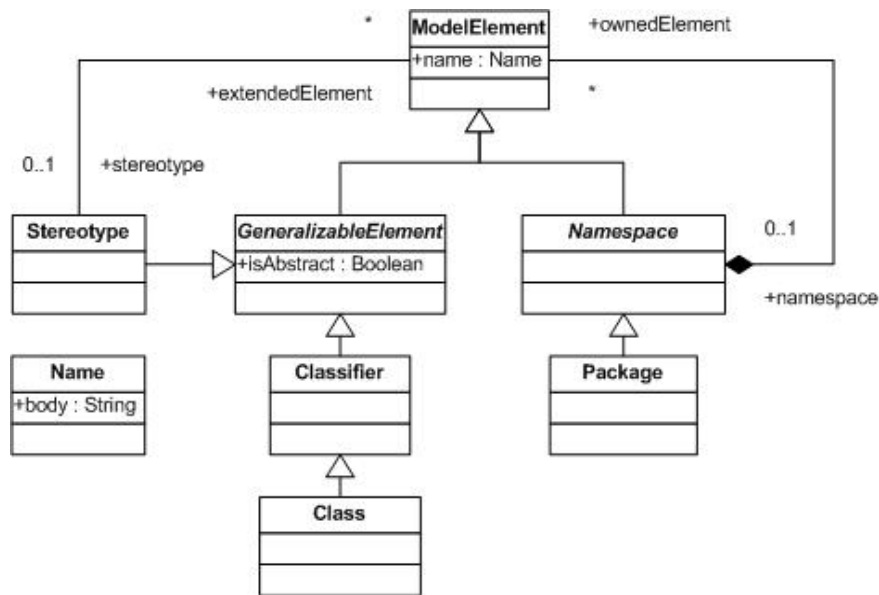


Рис. 3: Фрагмент метамодели 3.

```

context Association
inv: self.allConnections -> select(
    multiplicityRange -> select( upper > 1 ) -> notEmpty
).size < 2

```

5.3 Пример 3

Генератор базы данных использует для генерации пакеты, у которых имеется стереотип “Persistent”. Ограничение состоит в том, что все классы, содержащиеся в таких пакетах, тоже должны иметь стереотип “Persistent”. Фрагмент метамодели UML, иллюстрирующая это ограничение, приведена на рис. 3.

На языке OCL спецификация этого ограничения выглядит следующим образом:

```

context Package
inv: self.stereotype.name.body = 'Persistent'
    implies allOwnedElements -> forAll ( item: ModelElement |
        item.oclIsTypeOf( Class )
        implies item.stereotype.name.body = 'Persistent'
    )

```

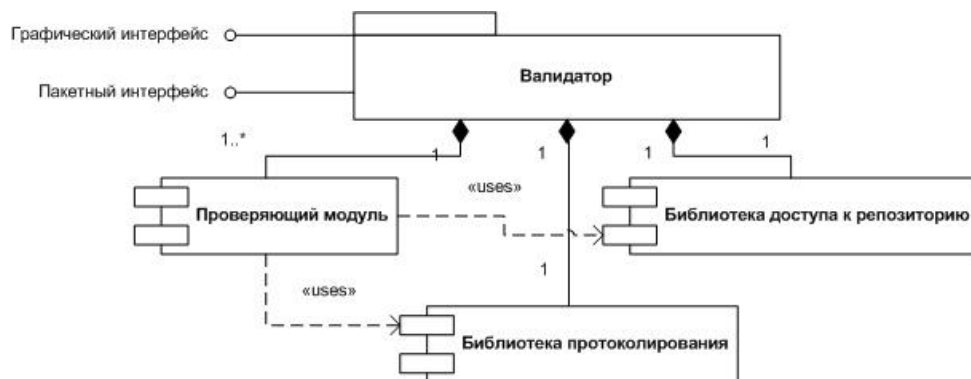


Рис. 4: Типовая архитектура валидатора

6 Валидатор визуальной спецификации проекта

На рис. 4 представлена архитектура типового валидатора визуальных моделей проекта. Основная часть валидатора — это *проверяющий модуль* (таких модулей может быть несколько). Он взаимодействует с репозиторием CASE-средства (или с XMI-файлом) через *библиотеку доступа к репозиторию*. Создание отчёта осуществляется при помощи специальной *библиотеки протоколирования*. Наконец, в валидатор так же входят блоки, реализующие интерфейсы управления — интерактивный пользовательский (графический) и пакетный.

7 Генератор валидаторов

Общий принцип действия генератора представлен на рис. 5. Исходный текст ограничений на OCL подвергается синтаксическому и видозависимому анализу, при этом используется информация о метамодели UML. Далее, генерируется код проверяющего модуля на языке Microsoft Visual Basic с учётом структуры репозитория CASE-пакета, для которого создаётся валидатор. Последнее обстоятельство важно ввиду того, что большинство промышленных CASE-пакетов, несмотря на заявления о полной поддержке UML, реализуют различные его подмножества, и поэтому при генерации конечного кода требуется преобразовывать ограничения на элементы метамодели в ограничения на элементы репозитория. Полученный модуль компилируется и компонуется с библиотеками протоколирования и доступа к репозиторию (см. рис. 4).

В качестве OCL-парсера использован пакет Dresden OCL Parser Toolkit [17].

Важно отметить, что в предлагаемом подходе генерируется конечный

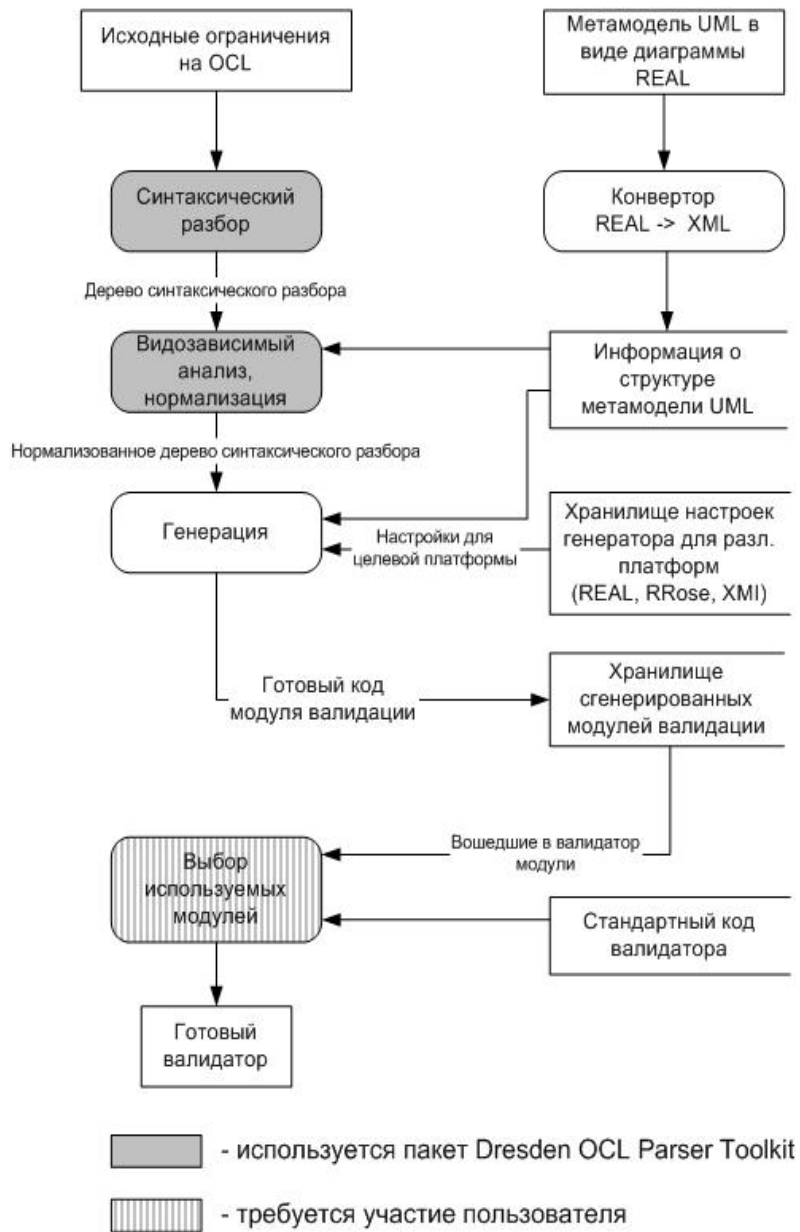


Рис. 5: Общий принцип работы генератора валидаторов.

выполнимый модуль, в отличие, например, от [29]. Это обеспечивает значительный выигрыш в производительности по отношению к интерпретационным решениям, так как OCL является декларативным языком и интерпретация его достаточно трудоёмка. Выигрыш в быстродействии можно использовать при встраивании валидатора в регулярную ночную сборку и регрессионное тестирование проекта, где время выполнения каждого этапа сборки критично.

8 Апробация

В рамках данной технологии создан валидатор для описанного выше телекоммуникационного проекта, предоставленного ЗАО “ЛАНИТ-ТЕРКОМ”. В проекте использовался CASE-пакет REAL, метамодель которого совмещает в себе UML и SDL, поэтому ограничения накладывались не на метамодель UML, а на метамодель REAL [9]. В валидаторе была реализована проверка 11 различных OCL-ограничений (40 строк OCL-кода). Время его работы — 30-40 минут на Intel Celeron-900 MHz. С помощью валидатора было выявлено 16 ошибок. Валидатор работал в пакетном режиме, автоматически запускаясь каждую ночь.

По итогам апробации были сделаны следующие выводы относительно случаев эффективного использования валидатора:

1. При контроле ограничений, которые либо не больше нигде не проверяются автоматически, (например, ограничения по стандартному оформлению диаграмм), либо проверяются лишь на поздних этапах (например, как ограничения на именование идентификаторов, нарушение которых будет выявлено только при компиляции сгенерированного кода).
2. В случае, когда автоматической обработке визуальной модели предшествует длительный процесс собственно моделирования. В этой ситуации имеет смысл как можно раньше наладить base-line [27], и поддерживать её в том числе с помощью регулярно запускаемого в пакетном режиме валидатора (в стиле регулярной сборки [28]). Иначе, как показывает опыт, установка процесса генерации, компиляции сгенерированного кода и его отладки становится очень трудоёмкой.
3. В случае, когда при разработке в модель вносятся крупные изменения, например, вследствие реализации дополнительной функциональности. В этой ситуации изменения могут вноситься не автором соответствующего фрагмента модели, поэтому оказывается важным как можно раньше контролировать случаи нарушения целостности визуальной модели.

9 Заключение

К настоящему моменту реализована поддержка большей части конструкций OCL и взаимодействие с CASE-пакетом REAL. В дальнейшем планируется полная реализация поддержки OCL и интеграция с пакетом Rational Rose, а так же поддержка формата XMI. Планируется также провести эксперименты с интерактивным интерфейсом валидатора, созданием сценариев диалоговой работы с пользователями CASE-пакетов.

Список литературы

- [1] Д.В. Кознов: Визуальное моделирование компонентного программного обеспечения. // Диссертация на соискание ученой степени кандидата физико-математических наук, Санкт-Петербург, 2000. 82 с.
- [2] <http://www.omg.org/uml/>
- [3] Д.В. Кознов: Коммуникативный аспект визуального моделирования при разработке программного обеспечения. // Тезисы III съезда психологов России, Санкт-Петербург, Июнь 2003.
- [4] D.Boulychev, D.V.Koznov, A.A.Terekhov: On project-specific languages and their application in reengineering. // In Proceedings of Conference on Software Maintenance and Reengineering, Budapest, Hungary, 11-13 March 2002.
- [5] Лотман Ю. М.: Текст в тексте // Учен. зап. Тартуского ун-та, 1981. – Вып. 567.
- [6] <http://vai.vom/products/alm/erwin.htm>
- [7] Парфенов В.В., Терехов А.Н.: RTST - технология программирования встроенных систем реального времени. // Системная информатика. Вып. 5: Архитектурные, формальные и программные модели. – Новосибирск, 1997, с. 228-256.
- [8] С.А.Стригун. Опыт использования технологии REAL для создания информационных систем. Дипломная работа. Математико-Механический факультет, СПбГУ, 2001.
- [9] А.Н.Терехов К.Ю.Романовский, Дм.В.Кознов, П.С.Долгов, А.Н.Иванов: Real: Методология и CASE-средство для разработки систем реального времени и информационных систем // Программирование, 1999, N 5.
- [10] Egyed, A. Heterogeneous View Integration and its Automation. // Technical Report, Center for Software Engineering, University of Southern California, Los Angeles, CA 90089-0781, April 1999.

- [11] Aliko Tsiolakis. Integrating Model Information in UML Sequence Diagrams. // In Proc. of the Satellite Workshops of the 28th ICALP (GT-VMT 2001), Electronic Notes in Theoretical Computer Science 50 No. 3, July 2001.
- [12] Aliko Tsiolakis. Semantic Analysis and Consistency Checking of UML Sequence Diagrams. // Diplomarbeit, Technische Universität Berlin, Technical Report No. 2001/06, April 2001.
- [13] Aliko Tsiolakis. Consistency Analysis of UML Class and Sequence Diagrams based on Attributed Typed Graphs and their Transformation. // Technische Universität Berlin, Technical Report No. 2000/3, March 2000.
- [14] A. Tsiolakis, H. Ehrig. Consistency Analysis of UML Class and Sequence Diagrams using Attributed Graph Grammars. // In Proc. GRATRA 2000 (Ehrig, Taentzer Eds.), TU Berlin, FB Informatik, TR No. 2000-2, pp. 77-86, March 2000.
- [15] Steve Cook and John Daniels. Designing Object Systems: Object-Oriented Modelling with Syntropy // Prentice Hall, 1994.
- [16] <http://www-3.ibm.com/software/awdtools/library/standards/ocl.html>
- [17] Heinrich Hussmann, Birgit Demuth, and Frank Finger. Modular architecture for a toolset supporting OCL. // In Andy Evans, Stuart Kent, and Bran 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.
- [18] <http://argouml.tigris.org>
- [19] Wolfgang Ahrendt, Thomas Baar, Bernhard Beckert, Richard Bubel, Martin Giese, Reiner Hoehnle, Wolfram Menzel, Wojciech Mostowski, Andreas Roth, Steffen Schlager, and Peter H. Schmitt. The Key Tool. // Department of Computing Science, Chalmers University and Goeteborg University, Technical Report in Computing Science No. 2003-5, February 2003.
- [20] Laurent Monestel, Tewfik Ziadi, and Jean-Marc Jezequel. Product line engineering: Product derivation. // In Workshop on Model Driven Architecture and Product Line Engineering, associated to the SPLC2 conference, San Diego, August 2002.
- [21] L.M.Norton. SEI's Software Product Line Tenets. // IEEE Software, July/August 2002, pages 32-41.
- [22] Wai-Ming Ho, Jean-Marc Jezequel, Alain Le Guennec, and Francois Penaneac'h. UMLAUT: an extendible UML transformation framework. // In Proc. Automated Software Engineering, ASE'99, Florida, October 1999.
- [23] Objectteering Software, <http://www.objectteering.com/>

- [24] <http://lci.cs.ubbcluj.ro/ocle/index.htm>
- [25] ROCASE, <http://cs.ubbcluj.ro/rocase/>
- [26] Dan Chiorean, Dragos Cojocari. Implementation of OCL Support in UML CASE Tools - the ROCASE Experience; Objectives, Proposals, Perspectives. // In Proc. of 4th International Conference on Information Systems Modelling, ISM '01, Hradec nad Moravici, Czech Republic, May 2001.
- [27] Watts Humphrey, Managing the Software Process. // Addison-Wesley, 1989, 512 стр.
- [28] I. Sommerville, Software Engineering. 6th Edition. // Addison-Wesley, 2001. 963 pages // Русский перевод: И. Соммервилл. Инженерия программного обеспечения - М: "Вильямс", 2002 г, 624 стр.
- [29] Wuwei Shen, Kevin Compton, James K. Huggins. A Validation Method for a UML Model Based on Abstract State Machines // Proceedings of EUROCAST 2001, С. 220-223