Формальное моделирование спецификаций процессов для композиционного проектирования потоков работ

Н. А. Земцов, С. А. Ступников

Институт Проблем Информатики РАН
{nazem, ssa}@ipi.ac.ru

Аннотация. В данной статье рассматривается метод формального моделирования спецификаций процессов языка СИНТЕЗ при помощи Нотации Абстрактных Машин (AMN). Метод используется для формального обоснования корректности композиционного проектирования потоков работ.

1. Введение

Идеи и методы композиционного проектирования информационных систем (ИС) - проектирования ИС из существующих компонентов, несмотря на сложности их реализации, начинают играть все более значимую роль при построении ИС. Это связано в первую очередь с усложнением ИС в целом с одной стороны, и с достижениями в развитии информационной инфраструктуры с другой. Существуют задачи построения ИС, которые не могут быть адекватно решены с помощью существующих технологий проектирования «сверху вниз». Примером области, иллюстрирующей данное суждение, является EAI (Enterprise Application Integration), где возникает необходимость интеграции ИС нескольких предприятий с помощью глобальной сети для совместной работы. Другим примером (частично пересекающимся с первым) является область электронной коммерции (e-business). В области Web-приложений существует тенденция к увеличению использования технологии Web-сервисов [webservices]. Web-сервисы разрабатываются и предоставляются множеством компаний, так что становится возможным говорить о web-сервисах как о существующих компонентах.

Одной из самых широко распространенных технологий построения ИС предприятий, как традиционных, так и Web-ориентированных, является технология потоков работ (workflows), возникшая в области автоматизации документооборота на предприятиях. Сдерживающим фактором для развития и повсеместного применения композиционного проектирования в данной области является большое разнообразие различных моделей (например, WSDL [wsdl], OWL-S [owls], BPEL4WS [bpel]) которые не интегрированы между собой.

Метод композиционного проектирования ИС из различных компонентов (информационных, программных, процессных), целенаправленно разрабатывается в Лаборатории методов композиционного проектирования ИС ИПИ РАН [adbis98]. В качестве канонической информационной модели, предназначенной для унифицированного представления спецификаций компонентов, метод ис-

пользует язык СИНТЕЗ [synth], ориентированный на семантическую интероперабельность и композиционное проектирование информационных систем в широком диапазоне существующих неоднородных информационных ресурсов. Этот язык обладает гибридными возможностями, обеспечивающими интеграцию как структурированных, так и слабо структурированных моделей данных. Одна из разновидностей моделей процессов языка СИНТЕЗ, предназначенная для спецификации потоков работ, основана на процессной алгебре CSP – Communicating Sequential Processes [csp] (именно процесс, как некоторая последовательность действий, преследующая определенную цель, является основной структурной единицей потока работ [wfmc]).

Общая схема композиционного метода проектирования ИС состоит в следующем. Первичное проектирование ИС осуществляется при помощи стандартных методов Объектного Анализа и Проектирования [oaddescr]. Описание ИС, полученное на данном этапе, представляется спецификацией на одном из широко используемых языков (таких как UML, WSDL и пр.) или на их совокупности. Различные существующие компоненты (например, web-сервисы) также должны быть представлены спецификациями на некоторых языках. В целях унифицированного представления спецификаций, языки, использующиеся при первичном проектировании, а также для описания компонентов, отображаются в язык СИНТЕЗ. Примеры подобных отображений для XML Schema и WSDL описаны в [wsdl2synth, xmls2synth]. Следующим этапом композиционного проектирования является поиск релевантных компонентов, т.е. таких, из которых требуемая ИС могла бы быть построена. Релевантные компоненты соединяются в композицию в рамках исчисления спецификаций, предложенного в [compcalculus].

Завершающим этапом композиционного проектирования является формальное доказательство корректности построения композиции. Условием корректности является уточнение спецификации требуемой ИС спецификацией композиции компонентов.

Понятие уточнения (refinement) ИС исследовалось и формализовывалось в течение многих лет [bbook, back]. Неформально, система B уточняет систему A, если разработчик может использовать систему B вместо системы A, не замечая разницы. Описываемый метод композиционного проектирования использует в качестве языка формализации понятия уточнения Нотацию Абстрактных Машин (Abstract Machine Notation, AMN). AMN представляет собой формальный язык спецификаций, основанный на логике предикатов первого порядка и теории множеств. Для AMN разработана специальная технология (B-technology) и комплекс программных средств (B-Toolkit [bmanual]), позволяющие доказывать факт уточнения при помощи автоматических и интерактивных средств доказательства.

Данная статья[1] описывает метод моделирования процессов языка СИНТЕЗ средствами AMN, составляющий формальную базу завершающего этапа композиционного проектирования. Отображение спецификаций процессов языка СИНТЕЗ в AMN позволяет использовать B-технологию для доказательства корректности уточнения спецификации требований композицией спецификаций компонентов. Корректность отображения доказывается с использованием методов операционной семантики.

[1] Работа частично поддержана РФФИ, грант 03-01-00821

Статья структурирована следующим образом. В разделе 2 описывается модель процессов языка СИНТЕЗ. В разделе 3 на примере излагаются основные принципы отображения модели процессов СИНТЕЗ в AMN. В разделе 4 на примере демонстрируется применение отображения спецификаций процессов СИНТЕЗ в абстрактные машины AMN для доказательства корректности композиционного проектирования ИС. В разделе 5 кратко описаны основные работы, родственные данной. В заключении суммируются результаты работы.

2. Модель процессов языка СИНТЕЗ

Язык СИНТЕЗ ориентирован на семантическую интероперабельность и композиционное проектирование ИС в широком диапазоне существующих неоднородных информационных ресурсов. Этот язык обладает гибридными возможностями, обеспечивающими интеграцию как структурированных, так и слабо структурированных моделей данных. В частности, язык включает унифицированную систему типов: универсальный конструктор произвольных абстрактных типов данных (АТД), а также представительный набор встроенных типов. АТД может содержать атрибуты, сигнатуры методов и процессное описание. Нижеприведенный пример описывает АТД `complexTransmitter` – передатчик, содержащий методы приема, проверки, коррекции и посылки некоторого сообщения, а также процессное описание.

```
{ complexTransmitter;
in: type;
receive: { in: function; };
verify_head: { in: function; };
verify_tail: { in: function; };
correct: { in: function; };
send; { in: function; };
process:
{ IdleR; { receive -> ( (VerHead) || (VerTail) ) >> (Check) } },
{ VerHead; { verify_head -> skip } },
{ VerTail; { verify_tail -> skip } },
{ Check; { correct -> send -> IdleR } }
}
```

Основой процессной модели языка СИНТЕЗ является процессная алгебра CSP [csp]. Корректное процессное описание на языке СИНТЕЗ представляет собой строку в грамматике, состоящую из последовательности подстрок, соответствующих именованным процессным термам (NamedProcess). Процессное описание накладывает ограничения на порядок вызова методов следующим образом. При создании объекта абстрактного типа данных с процессным описанием, процесс инициализируется первым в спецификации именованным процессом. После этого на протяжении всего времени существования данного объекта могут быть вызваны только те его методы, которые разрешаются процессной логикой (это – те, которым соответствуют события из процессного описания, разрешенные текущим состоянием процесса). Например, сразу же после

создания объекта типа `complexTransmitter`, может быть вызван метод получения закодированного сообщения – `receive`. Затем первая часть сообщения (head) и вторая часть (tail) проверяются на соответствие шаблону используемого кода методами `verify`. Проверка первой и второй части сообщения производится параллельно. Далее метод `correct`, если это нужно и возможно, производит коррекцию сообщения; затем сообщение посылается методом `send`.

Синтаксис допустимого (для разработанного отображения в AMN) процессного описания в форме Бэкуса-Наура (BNF) выглядит следующим образом.

```
ProcessDescription ::= NamedProcess << , NamedProcess>>
NamedProcess ::= {ProcessName; {GeneralProcess} }
GeneralProcess ::= ChoiceProcess |
 (ChoiceProcess) '>>' (ProcessCall)
ChoiceProcess ::= ChoiceCombined |
 (ChoiceCombined) [] (ChoiceCombined) <<[] (ChoiceCombined)>>
ChoiceCombined ::=Prefixing|
 if expr then (ChoiceCombined) < else (ChoiceCombined)>
Prefixing ::= event -> Operation | event -> ProcessCall
Operation ::=
  (ProcessCall) * (ProcessCall) << * (ProcessCall)>> |
  (ProcessCall) || (ProcessCall) << || (ProcessCall)>> |
  (ProcessCall) |[events]| (ProcessCall)
ProcessCall ::= ProcessName | skip | stop | GeneralProcess
ProcessName ::= identifier
events ::= Event <<, Event>>
Event ::= identifier
expr ::= произвольный предикат AMN
identifier ::= последовательность из букв и цифр, первый символ - буква
```

Вертикальная черта elem1 | elem2 в грамматике означает выбор между elem1 и elem2. Одинарные скобки <elem> означают повторение elem 0 или 1 раз. Двойные скобки <<elem>> означают повторение elem произвольное число раз, включая нуль.

Приведенной грамматике соответствуют процессные термы, построенные на основание следующего представительного множества операций CSP:

- Префиксирование *a->Q*. Процесс ожидает вызова метода *a*, а после вызова метода передает управление процессу *Q*.
- Детерминированный выбор *a->P[]b->Q*. Процесс ожидает вызова метода *a*, либо вызова метода *b*, и в зависимости от того, вызов какого метода произошел, передает управление процессу *P* либо процессу *Q*.
- Процесс *STOP*. Состояние тупика – никакой метод объекта не может быть вызван.
- Процесс *SKIP*. Успешное завершение процесса. Управление передается процессу, находящемуся с текущим процессом в последовательной композиции.
- Последовательная композиция *P>>Q*. В случае успешного завершения процесса *P,* управление передается процессу *Q*.
- Условный оператор *if exp then P else Q*. Если значение выражения *exp* есть *true*, управление предается процессу *P*, иначе – процессу *Q*.

- Параллельная композиция $P||Q$. Процессы P и Q исполняются параллельно.
- Недетерминированный выбор $P*Q$. Исполняется процесс либо P, либо Q, выбор определяется системой.

3. Отображение модели процессов в AMN

AMN представляет собой формальный язык спецификаций, основанный на логике предикатов первого порядка и теории множеств. Основной единицей спецификации в AMN является абстрактная машина – примерный аналог абстрактного типа данных в языках программирования. В качестве примера рассмотрим машину, полученную в результате отображения процессного описания АТД complexTransmitter, описанного в разделе 2, в AMN.

```
MACHINE complexTransmitterProcess
SETS States = {IdleR, Skip, ReadyToSend, RemTail, Check, RemHead,
NotActive}
VARIABLES Main, Sub1, Sub2
INVARIANT
  Main : States & Sub2 : States & Sub1 : States
INITIALISATION
  Main := IdleR || Sub1 := NotActive || Sub2 := NotActive
OPERATIONS
receive =
 SELECT Main=IdleR & Sub1=NotActive & Sub2=NotActive
 THEN Main:=NotActive || Sub1:=RemHead || Sub2:=RemTail
 END;
verify_head = SELECT Sub1=RemHead THEN Sub1:=Skip END;
verify_tail = SELECT Sub2=RemTail THEN Sub2:=Skip END;
Success =
 SELECT Main=NotActive & Sub1=Skip & Sub2=Skip
 THEN Main:=Check || Sub1:=NotActive || Sub2:=NotActive
 END;
correct = SELECT Main=Check THEN Main:=ReadyToSend END;
send = SELECT Main=ReadyToSend THEN Main:=IdleR END
END
```

Абстрактная машина имеет секцию множеств (SETS), секцию переменных (VARIABLES), инвариант (INVARIANT), в котором переменные типизируются и связываются ограничениями, секцию инициализации переменных (INITIALISATION), а также секцию операций (OPERATIONS). Операции могут изменять состояние переменных машины.

Основные принципы отображения процессов СИНТЕЗ в AMN могут быть продемонстрированы на примере отображения процессного описания АТД complexTransmitter в абстрактную машину complexTransmitterProcess.

Процесс, задаваемый процессным описанием в АТД, понимается как система с конечным числом состояний; переходы из состояния в состояние осуществляются при вызове методов АТД. При моделировании процесса в AMN множе-

ство состояний системы описывается явным образом, методы АТД моделируются операциями абстрактной машины.

Множество состояний `States` процесса, а также его подпроцессов формируется на основании синтаксического анализа процессного описания. Также на основании синтаксического анализа определяется количество переменных, необходимых для хранения текущего состояния процесса. В случае `complexTransmitter` таких переменных три: `Main`, отвечающая основному процессу приема-передачи, `Sub1` и `Sub2`, отвечающие параллельным подпроцессам проверки частей сообщения на соответствие шаблону. Инициализация устанавливает переменную `Main` в состояние `IdleR` ожидания получения сообщения, а переменные `Sub1`, `Sub2` – в состояние `NotActive`, говорящее о том, что подпроцессы проверки изначально неактивны.

Тела операций абстрактной машины представляют собой обобщенные подстановки (generalized substitutions, [bbook]) вида

SELECT *G* THEN *S* END

где *G* представляет собой некоторое условие на переменные, а *S* – оператор, изменяющий переменные. Например, подстановка, составляющая тело операции `correct`,

SELECT *Main=Check* THEN *Main:=ReadyToSend* END

в случае, если основной процесс приема-передачи, задаваемый переменной `Main`, находится в состоянии `Check`, переводит основной процесс в состояние `ReadyToSend`. Каждому методу АТД `complexTransmitter` соответствует операция абстрактной машины. Кроме того, в машине присутствует особая операция `Success`, отвечающая успешному завершению подпроцессов. Эта операция переводит параллельные подпроцессы проверки в неактивное состояние и переводит основной процесс приема-передачи из неактивного состояния в состояние готовности к коррекции `Check`.

В рамках проекта по разработке композиционного метода проектирования в Лаборатории методов композиционного проектирования ИС ИПИ РАН были разработаны алгоритмы отображения спецификаций процессов языка СИНТЕЗ, синтаксис которых удовлетворяет грамматике, описанной в разделе 2, в AMN. Корректность отображения процессных спецификаций в AMN была доказана с использование методов операционной семантики [plotkin, opsem] по следующей схеме. Процессному описанию на языке СИНТЕЗ сопоставляется семантика в терминах Систем Помеченных Переходов (Labelled Transition System, LTS) [roscoe]. Семантика в терминах LTS также сообщается абстрактным машинам AMN. Сохранение семантики при отображении процессного описания на языке СИНТЕЗ в AMN, доказывается индукцией по построению процессного описания.

Разработанные алгоритмы были реализованы на языке Java и составили основу программного средства SPM2B (SYNTHESIS Process Model to B), автоматически осуществляющего отображение спецификаций процессов на языке СИНТЕЗ в AMN.

4. Доказательство уточнения для спецификаций процессов

Продемонстрируем на примере применение отображения спецификаций процессов СИНТЕЗ в абстрактные машины AMN для доказательства корректности уточнения спецификации требований композицией спецификаций компонентов при композиционном проектировании ИС. Рассмотрим пример фрагмента спецификации требований – АТД Transmitter. Данный тип специфицирует передатчик, последовательно принимающий и передающий некоторые кодированные сообщения. Спецификация типа на СИНТЕЗ включает методы `receive`, `send` и процессное описание, регламентирующее только поочередный порядок вызова методов приема и передачи сообщений.

```
{ Transmitter; in: type;
   receive; { in: function; };
   send; { in: function; };
   process:
    { Idle; { receive -> Remember } },
    { Remember; { send -> Idle } };
}
```

Предположим, что среди спецификаций существующих компонентов присутствует спецификация типа `complexTransmitter` (приведенная в разделе 2), которая не совпадает в точности со спецификацией требований. Тем не менее, если удастся доказать факт уточнения между ними, то компонент `complexTransmitter` можно будет использовать в качестве `Transmitter` при построении ИС. Для формального доказательства уточнения спецификации типа `Transmitter` спецификацией типа `complexTransmitter` эти спецификации отображаются в AMN при помощи программного инструмента SPM2B. Результирующая AMN-спецификация процесса `complexTransmitter` приведена в разделе 3, AMN-спецификация процесса `Transmitter` имеет вид

```
MACHINE TransmitterProcess
SETS TPStates = {Remember, Idle}
VARIABLES Proc
INVARIANT Proc : TPStates
INITIALISATION Proc := Idle
OPERATIONS
 receive = SELECT Proc=Idle THEN Proc:=Remember END;
 send = SELECT Proc=Remember THEN Proc:=Idle END
END
```

Эксперт-конструктор принимает участие в процессе доказательства уточнения следующим образом. Во-первых, эксперт формирует инвариант уточнения – условия взаимосвязи состояний уточняемой машины `TransmitterProcess` и состояний уточняющей машины `complexTransmitterProcess` и добавляет его в секцию `INVARIANT` машины `complexTransmitterProcess`:

```
( (ST01=Idle) <=>
```

```
    (Main=IdleR & Sub1=NotActive & Sub2=NotActive) ) &
  ( (ST01=Remember) <=>
    (Main:{NotActive, Check, ReadyToSend}) &
    (Sub1:{VerHead, Skip, NotActive}) &
    (Sub2:{VerTail, Skip, NotActive}) )
```

Во-вторых, эксперт подает AMN-спецификации `TransmitterProcess` и `complexTransmitterProcess` на вход программного средства B-Toolkit. По паре спецификаций B-Toolkit генерирует набор условий уточнения (proof obligations). Условия уточнения могут быть частично доказаны автоматически при помощи B-Toolkit, частично условия доказываются экспертом с помощью средства интерактивного доказательства теорем. Факт доказательства всех условий уточнения означает, что компонент, описываемый типом `complexTransmitter`, может быть использован при конструировании ИС вместо компонента, описываемого типом `Transmitter`.

Родственные работы

Методы моделирования спецификаций процессов средствами AMN разрабатывались в течении ряда лет различными исследовательскими группами.

Работа Абриаля [abrial2000] нацелена на усовершенствование схемы формальной разработки программного обеспечения. Обычно такая схема подразумевает разработку программы начиная с абстрактной формальной спецификации серией формально уточняющих шагов к конечной программе. Усовершенствованная схема предполагает выделение в спецификации частей, которые могут уточняться независимо. Такие части называются событиями. Конструкции, определяющие последовательность событий, отделяются от конструкций, описывающих события. Таким образом, средствами AMN фактически моделируется и уточняется процесс. Средства спецификации процессов, доступные при таком подходе, включают префиксирование, детерминированный выбор, условный оператор и рекурсию.

Работа Батлера [csp2b] направлена на сопряжение возможностей AMN и процессной алгебры CSP при спецификации ИС. Программное средство csp2B транслирует CSP-подобные спецификации процессов в абстрактные машины AMN. Средства спецификации процессов, транслируемые при помощи csp2B, включают префиксирование, детерминированный выбор, условный оператор, рекурсию, параллельность на внешнем уровне и процесс STOP, описывающий состояние тупика. Данный подход позволяет также накладывать ограничения на порядок исполнения операций стандартной абстрактной машины и тем самым органично дополнять AMN возможностью совместной спецификации и уточнения как алгоритмических аспектов ИС (выражаемых событиями), так и процессных аспектов ИС.

Работа Treharne и Schneider [treharne] также направлена на сопряжение двух взглядов на ИС. Первый из взглядов, традиционный для B-технологии, состоит в понимании спецификации системы как набора операций и анализа эффекта отдельных операций. Второй взгляд, характерный для процессных алгебр, в частности CSP, состоит в понимании спецификации системы как шаблона после-

довательности событий. Средства спецификации процессов, включающие префиксирование, выбор, рекурсию, условный оператор и STOP, используются в данной работе для описания так называемого управляющего цикла (control executive), определяющего порядок вызова операций абстрактной машины. Главным результатом работы является формальное определение условия отсутствия тупика в процессном описании.

Новизна метода моделирования средств спецификации процессов в AMN, разработанного в Лаборатории методов композиционного проектирования ИС, по сравнению с вышеупомянутыми работами выражается в поддержке таких операций конструирования процессов, как недетерминированный выбор, параллельная композиция и синхронизация на произвольном уровне описания (а не только на внешнем, как это делается в [csp2B]), последовательная композиция и процесс SKIP. Необходимость использования перечисленных операций для моделирования реальных процессов как составных частей потоков работ подробно обосновывается в [wfpatterns].

Заключение

Метод моделирования спецификаций процессов языка СИНТЕЗ средствами AMN, иллюстрируемый в данной статье, является формальной базой завершающего этапа метода композиционного проектирования ИС – доказательства корректности уточнения спецификации требований композицией спецификаций компонентов. Метод позволяет использовать программное средство B-Toolkit для автоматического и интерактивного доказательства факта уточнения.

Разработаны алгоритмы, осуществляющие отображение в AMN таких средств конструирования процессов, как недетерминированный выбор, параллельная композиция и синхронизация на произвольном уровне описания, последовательная композиция и процесс SKIP. При помощи методов операционной семантики доказана корректность разработанных алгоритмов.

Разработанные алгоритмы реализованы в виде программного средства SPM2B, осуществляющего автоматическое отображение спецификаций процессов языка СИНТЕЗ в AMN. В дальнейшем планируется исследование возможности применения описанного метода для более широкого класса моделей процессов.

Литература

[abrial2000] J.-R. Abrial. Event Driven Sequential Program Construction. Technical Report, Version 12, March 2000

[adbis98] Briukhov D.O., Kalinichenko L.A. Component-Based Information Systems Development Tool Supporting the SYNTHESIS Design Method. In *Proc. of the East European Symposium on "Advances in Databases and Information Systems"*, Poland, Springer, LNCS No.1475, 1998

[adbis2002] S.A. Stupnikov, L.A. Kalinichenko, J.S. DONG. Applying CSP-like Workflow Process Specifications for their Refinement in AMN by Pre-existing Workflows. In Proceedings of the Sixth East-European Conference on Advances in Databases and Information Systems ADBIS'2002, September 8-11, 2002, Bratislava, Slovakia

[back] Back R.-J., von Wright J. *Refinement Calculus: A systematic Introduction*. Springer Verlag, 1998
[bbook] J.-R. Abrial. The B-Book. Cambridge University Press, 1996.
[bmanual] B-Toolkit User's Manual - Release 3.4. Copyright B-Core (UK) Ltd., 1997.
[bpel] Tony Andrews et al. Business Process Execution Language for Web Services Version 1.1 http://www.ibm.com/developerworks/library/ws-bpel/, May 2003.
[comp-calculus] Kalinichenko L. A. Compositional Specification Calculus for Information Systems Development. In *Proc. of the East-West Symposium on Advances in Databases and Information Systems (ADBIS'99)*, Maribor, Slovenia, September 1999, Springer Verlag, LNCS, 1999
[csp] Ч. Хоар. Взаимодействующие последовательные процессы. М.: Мир, 1989.
[csp2b] M.J. Butler. csp2B: A Practical Approach To Combining CSP and B. Declarative Systems and Software Engineering Group, Technical Report DSSE-TR-99-2, February 1999.
[oaddescr] *Object Analysis and Design: Description of Methods* / Hutt A. (editor) // Object Management Group, John Wiley and Sons, 1994
[opsem] S.D. Brookes, A.W. Roscoe, and D.J. Walker. An Operational Semantics for CSP. Unpublished report.
[owls] OWL-S: Semantic Markup for Web Services The OWL Services Coalition. http://www.daml.org/services/owl-s/1.0/owl-s.pdf
[plotkin] Gordon D. Plotkin. A Structural Approach to Operational Semantics. Aarhus University Report, 1981.
[roscoe] Roscoe, A. W. The Theory and Practice of Concurrency. Prentice-Hall, 1998
[synth] Л.А. Калиниченко. СИНТЕЗ: Язык определения, проектирования и программирования интероперабельных сред неоднородных информационных ресурсов. М.: ИПИ РАН, 1993.
[treharne] H. Treharne, S. Schneider. Using a Process Algebra to control B OPERATIONS. In K. Araki, A. Galloway and K. Taguchi, editors, IFM'99, York, Springer, 1999.
[webservices] G. Alonoso, F. Casati, H. Kuno, V. Machiraju “Web Services” Springer-Velag, Berlin, Heidelberg, 2004.
[wfmc] WFMC. Workflow Management Coalition Terminology and Glossary (WFMC-TC-1011). Technical report, Workow Management Coalition, Brussels, 1996.
[wfpatterns] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski & A.P. Barros, Workflow Patterns, Distributed and Parallel Databases, 14(3):5-51, 2003
[wsdl] Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl, 2001.
[wsdl2synth] Briukhov D.O., Kalinichenko L.A., Tyurin I.N. Extension of Compositional Information Systems Development for the Web Services Platform. In Proceedings of the Seventh East-European Conference on Advances in Databases and Information Systems ADBIS'2003, 2003, Dresden, Germany
[xmls2synth] Briukhov D.O., Tyurin I.N. Mapping XML Schema Data Types into SYNTHESIS Types. In *Proc. of the Russian Conference “Digital Libraries: Advanced Methods And Technologies, Digital Collections” (RCDL’2002)*, Dubna, 2002