OSTIS-2013
(Open Semantic Technologies for Intelligent Systems)

УДК 004.89

ТЕХНОЛОГИЯ ФОРМАЛЬНО-ЛОГИЧЕСКОГО ПРОЕКТИРОВАНИЯ И ПРОТОТИПИРОВАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

Ануреев И.С., Атучин М.М.

Институт систем информатики имени А.П. Ершова,
г. Новосибирск, Россия
anureev@iis.nsk.su
atuchin.m@gmail.com

В работе представлена технология, позволяющая выбирать оптимальную концептуальную конфигурацию интеллектуальной системы на этапе ее проектирования и создавать ее прототип на базе формально-логических методов. В частности, использование формально-логического аппарата обеспечивает дедуктивную верификацию свойств безопасности проектируемой интеллектуальной системы. Технология иллюстрируется на примере спецификации концептуальной модели системы поддержки принятия решений.
Ключевые слова: операционно-онтологическая семантика, логика безопасности, системы переходов, интеллектуальные системы.

Введение

Применение формально-логических методов для обеспечения безопасности и эффективности программного обеспечения становится стандартом при проектировании и прототипировании индустриальных программных систем (прежде всего критического программного обеспечения, примерами которого являются бортовые спутниковые системы, встроенные автомобильные системы, банковские системы, системы проведения выборов и т. п.). Наметившаяся в последнее время тенденция использования интеллектуальных компонент в таких системах делает актуальной задачу разработки соответствующих формальных методов для интеллектуальных систем (далее ИС) с учетом их специфики.

Следует также отметить, что в настоящее время происходит переход от разработки конкретных ИС к созданию технологий их разработки [Хорошевский, 2008; Хорошевский, 2009; Хорошевский, 2012а; Хорошевский, 2012б; Голенков, 2011; Грибова и др., 2012; Загорулько, 2009]. Поэтому важной задачей является создание технологии на базе формально-логических методов (далее ФЛП-технологии) применительно к задачам проектирования и прототипирования ИС.

В статье предлагается подход к решению этих двух задач.

1. Базис ФЛП-технологии

ФЛП-технология базируется на специализированных помеченных системах переходов (далее П-системах) [Anureev, 2012], ориентированных на ускоренную разработку средств спецификации и верификации программных систем (далее ПС).

П-система строится над двумя базовыми множествами. Первое множество, называемое множеством элементов, определяет всевозможные синтаксические и семантические объекты, которые могут использоваться в специфицируемой ПС. Второе множество, называемое множеством атомов, определяет наименьшие синтаксические объекты языка П-систем, используемого для описания функциональностей ПС, правил перехода, имен объектов ПС и т. д. Элементы и атомы вместе называются атоментами, а язык П-систем — языком Atoment.

Одним из требований к языку Atoment является его близость по синтаксису к естественному языку. Основными синтаксическими объектами языка Atoment являются последовательности, списки и выражения. Атомы соответствуют словам естественного языка. Последовательности объектов (атомов, элементов и т. д.) — это упорядоченные наборы объектов, разделенных пробелами. Последовательности атомов соответствуют фразам (последовательностям слов) естественного языка. Списки — это последовательности, заключенные в круглые скобки (и). Списки атомов соответствуют

предложениям естественного языка, а последовательности списков атомов — текстам на естественном языке. Выражения — это либо атомы, либо списки выражений. Выражения можно рассматривать как вид иерархического текста со сложной вложенной структурой (определяемой скобками).

Особенностью П-систем является использование алгебраических систем в качестве состояний. Выбор подходящей сигнатуры алгебраической системы при проектировании и прототипировании ПС позволяет в какой-то степени решать проблему моделирования концептуального окружения ПС (системы понятий и категорий, которая позволяет сделать спецификации ПС обозримыми и емкими в смысловом аспекте за счет переноса части смысла на уровень терминов и обозначений.).

В отличие от стандартных алгебраических систем, используемых, например, в машинах абстрактных состояниях [Gurevich, 2004], в которых символы сигнатуры — атомы, символы сигнатуры в П-системах — списки атомов. Использование списков атомов в качестве символов сигнатуры (далее символов) позволяет приблизить описание вызовов функций, обозначаемых этими символами, к описанию на естественном языке. Другой важной особенностью таких символов является использование в них специальных атомов _ и __, которые делают эти символы шаблонами для соответствующих функций, обеспечивая некоторую информацию о вызовах этих функций. Эти атомы, называемые спецификаторами аргументов, означают места для аргументов функций. Атом _, входящий в символ сигнатуры, дополнительно указывает на то, что аргумент при вызове соответствующей этому символу функции нужно вычислить, а атом __ — что вычислять аргумент не нужно. Например, символ `(_ is concept)` является шаблоном для логической функции, которая возвращает значение `true`, если ее (единственный) аргумент является понятием. Число вхождений спецификаторов аргументов в символ называется его местностью (и местностью функции, обозначаемой этим символом), а i-е вхождение спецификатора аргумента в этот символ — i-м аргументом соответствующей функции.

Символы сигнатуры делятся на модифицируемые символы (например, символ `(_ is concept)`), семантика (интерпретация) которых может меняться при переходе П-системы из состояния в состояние, и предопределенные символы (например, коньюнкция `(_ and _)` и универсальный квантор `(forall __ __)`) с постоянной семантикой.

Конфигурации, в которых может находиться П-система, определяются обычным образом, как пары, состоящие из метки и состояния. Метками в П-системах являются программы, представляющие собой последовательности выражений на языке Atoment.

В [Anureev, 2012] выделены три вида специализированных систем переходов.

Ориентированные на операционную семантику системы переходов (далее ОС-системы) предназначены для разработки формальных спецификаций компьютерных языков и операционной семантики ПС.

Ориентированные на логику безопасности системы переходов (далее ЛБ-системы) предназначены для разработки дедуктивных методов обеспечения безопасности ПС.

Онтологические системы переходов (далее ОНТ-системы) предназначены для разработки операционно-онтологической семантики языков программирования [Ануреев, 2009] и ПС [Ануреев, 2008]. Эти системы являются специальным видом ориентированных на операционную семантику систем переходов, в которых в множестве модифицируемых символов выделяются два подмножества: множество онтологических символов и множество символов экземпляризации. Онтологические символы специфицируют элементы (понятия, атрибуты, отношения и т. п.) онтологии ПС. Символы экземпляризации специфицируют связи элементов онтологии с экземплярами.

Для каждого вида П-систем определен интерпретатор правил переходов, что делает спецификации ПС, описываемые с помощью П-систем, выполнимыми.

Специфика ИС состоит в том, что ФЛП-технология использует для проектирования и прототипирования ИС последние два вида П-систем и практически не использует ОС-системы, поскольку ИС, как правило, имеют сложную онтологическую структуру.

2. Этапы проектирования и прототипирования интеллектуальной системы в рамках ФЛП-технологии

Проектирование и прототипирование ИС на основе ФЛП-технологии состоит из следующих этапов.

На первых трех этапах строится прототип ИС.

Этап 1. Описывается множество состояний П-системы, которая специфицирует ИС. Поскольку для спецификации ИС используются ОНТ-системы, это описание должно включать множество онтологических символов и множество символов экземпляризации.

Этап 2. Определяется предметно-ориентированный язык (далее ПО-язык) как подмножество языка Atoment, описывающий желаемую функциональность ИС. Каждой функциональности соответствует конструкция ПО-языка. Также на этом шаге определяется неформальная семантика конструкций ПО-языка.

Этап 3. На основе ОНТ-систем, разрабатывается

операционно-онтологическая семантика ПО-языка.

Поскольку операционно-онтологическая семантика ИС, построенная на базе ОНТ-систем, является выполнимой, она задает прототип ИС.

На остальных этапах выбирается оптимальная концептуальная конфигурация ИС и доказывается ее безопасность.

Этап 4. На этом этапе, называемом этапом тестирования прототипа ИС, моделируются различные последовательности функционирования ИС. Целью такого тестирования является нахождение ошибок проектирования ИС и выбор оптимальной концептуальной конфигурации ИС.

Этап 5. Выполняется экспертная оценка «правильности» и «оптимальности» проектирования ИС посредством анализа текста формальных спецификаций ИС (соответствующей ОНТ-системы). Возможными действиями эксперта на этом этапе являются слияние и разделение функциональностей ИС, упрощение или обобщение элементов состояния ИС и т. д.

Этап 6. Разрабатывается логика безопасности для ПО-языка, основанная на ЛБ-системах. Особенность этого класса П-систем состоит в том, что, как правило, большая часть правил логики безопасности совпадает с соответствующими правилами операционно-онтологической семантики ПО-языка.

Этап 7. Выполняется расстановка свойств безопасности в программе на ПО-языке, которая специфицирует ИС, и с помощью логики безопасности выполняется проверка безопасности ИС относительно расставленных свойств. Фактически осуществляется дедуктивная верификация ИС с генерацией и последующим доказательством условий корректности. Если все сгенерированные условия корректности истинны, то ИС безопасна.

Заметим, что на каждом этапе может происходить возвращение к предыдущим этапам. Например, определение правил операционной семантики может потребовать уточнение набора функциональностей ИС или ее состояния.

Кроме того, ФЛП-технология допускает многовариантную разработку прототипа ИС, при которой на каждом этапе рассматривается набор вариантов. На шаге 1 вариантами будут различные описания состояния ИС, на шаге 2 — различные наборы функциональностей ИС и комбинирующих их конструкторов, и т. д.

3. Пример проектирования СППР с помощью ФЛП-технологии

В этом разделе мы опишем идею спецификации концептуальной модели СППР СОМТИ [Загорулько и др., 2010] (далее КМ) на языке Atoment, проиллюстрировав первые три этапа ФЛП-технологии. Для экономии места считается, что читатель знаком с работой [Загорулько и др., 2010].

На первом этапе определяется состояние ОНТ-системы, специфицирующей КМ, как интерпретация онтологических символов и символов экземпляризации.

Множество онтологических символов, используемых в состояниях ОНТ-системы, включает 4 символа. Символ `(_ is concept)` определяет множество понятий. Символ `(_ is attribute of _)` определяет множество атрибутов понятий, а его типизированный вариант `(_ is attribute of _ of _)` определяет множество типизированных атрибутов понятий. Типом является третий аргумент этого символа. Символ `(_ inherits _)` определяет отношение наследования на понятиях.

Интерпретация онтологических символов определяется следующим образом. Каждая сущность, используемая в КМ, описывается соответствующим понятием, которое добавляется в состояние предопределенным (из стандартной библиотеки языка Atoment) выражением `(A ::= B)`, называемым модификацией символа или присваиванием. Это выражение меняет интерпретацию символа сигнатуры, которому соответствует вызов `A` на значение выражения `B` для аргументов, входящих в `A`. Например, добавление понятия `адаптер`, соответствующего сущности адаптер, осуществляющей взаимодействие СППР СОМТИ с окружением, определяется присваиванием `((адаптер is concept) ::= true)`, изменяющим интерпретацию символа `(_ is concept)` в точке `адаптер` на `true`.

Атрибуты сущности, используемой в КМ, определяются как атрибуты соответствующего понятия. Так, например, понятие `задача` имеет атрибуты `имя`, `параметры`, `подзадачи`, `порождает` и `реализация`. Присваивание `((имя is attribute of задача of string) ::= true)` добавляет атрибут `имя` типа `string` понятию `задача`. Аналогичным образом определяются остальные атрибуты понятия `задача`. Заметим, что выбрано такое определение состояния, при котором бинарные отношения, используемые в КМ, также моделируются атрибутами. Например, бинарное отношение `подзадача` на задачах моделируется атрибутом `подзадачи` понятия `задача`. Можно было бы напрямую моделировать отношения, \используемые в КМ, если расширить множество онтологических символов символом `(_ is relation on _ and _)`. В этом случае, отношение `подзадача` определялось бы присваиванием `((подзадача is relation on задача and задача) ::= true)`.

Таким образом, интерпретация онтологических символов, специфицирующая онтологию КМ,

определяется программой, которая является последовательностью присваиваний онтологическим символам. Как правило, такая онтология не меняется при функционировании ОНТ-системы.

Множество символов экземпляризации, используемых в состояниях ОНТ-системы, включает 3 символа. Символ (_ is _) определяет множество экземпляров (первый аргумент) понятия (второй аргумент). Символ (_ of _) определяет значение атрибута (первый аргумент) экземпляра понятия (второй аргумент). Его модифицированный вариант (_ of _ of _) определяет значение атрибута (первый аргумент) экземпляра (второй аргумент) понятия (третий аргумент). Он используется для разрешения конфликта в случае, когда экземпляр принадлежит нескольким понятиям, имеющим один и тот же атрибут.

На втором этапе определяется ПО-язык, специфицирующий функциональность КМ как некоторое конечное множество параметризованных выражений (далее ПО-выражений) языка Atoment. На третьем этапе определяются правила перехода ОНТ-системы, задающие семантику ПО-выражений.

Функциональность КМ определяется как объединение функциональности понятий КМ. Функциональность понятия есть множество ПО-выражений, которые содержат в качестве аргумента хотя бы один атомент, являющийся экземпляром этого понятия. Например, функциональность понятия адаптер включает три параметризованных выражения, специфицирующих действия получить задание, загрузить данные и выгрузить данные. Пусть I — экземпляр понятия адаптер. Выражение (получить задание :: I) запрашивает задание у операционной среды. Атоменты, входящие в выражение и следующие за спецификатором контекста ::, называются контекстами этого выражения. Семантика ПО-выражения меняется в зависимости от используемых в ней контекстов. Выражение (получить задание :: I) задается правилом перехода (if (получить задание :: I) var I then (assume (I is адаптер)) (modify ((: value) is задание))).

Правило перехода представляет собой последовательность выражений языка Atoment. Часть if правила содержит образец, с которым сопоставляется ПО-выражение. Часть var содержит последовательность переменных образца, которые получают значения при сопоставлении ПО-выражения с образцом. Часть then, называемая телом правила, представляет собой последовательность выражений (программу), которая выполняется (после замены переменных образца соответствующими значениями) в случае, если ПО-выражение соответствует образцу.

Тело правила для ПО-выражения (получить задание :: I) представляет собой последовательность из двух выражений. Выражение (assume (I is адаптер)) выполняется тогда и только тогда, когда I — экземпляр понятия адаптер. Выполнение этого выражения не меняет состояния ОНТ-системы. В общем случае, предопределенное выражение (assume A), называемое условием продолжения, выполняется тогда и только тогда, когда истинна формула A. В противном случае, данная ветвь выполнения (последовательность конфигураций в терминах систем переходов) считается фиктивной и происходит откат (backtracking) в предыдущую конфигурацию и выбор другого правила для сопоставления. Выражение (modify ((: value) is задание)) изменяет интерпретацию нульместного символа (value) таким образом, что новое его значение является заданием (экземпляром понятия задание). В общем случае, предопределенное выражение (modify A), называемое условием модификации, преобразует состояние ОНТ-системы таким образом, что в новом состоянии истинна формула A. При этом могут изменяться только те символы, которые входят в A и которым предшествует атом :.

Таким образом, это выражение описывает некоторое декларативное действие, т. е., действие, которое определяет результат (с помощью логической формулы) и не определяет, как этот результат был получен. Использование декларативных действий позволяет выполнять проектирование на любом необходимом уровне абстракции, подбирая оптимальную «смесь» операционного и декларативного представлений. Выразительность декларативного представления определяется логикой (пропозициональная логика, логика предикатов, логика высшего порядка, табличная логика вида условие–результат, используемая при описании требований к проектируемой ИС и т. д.), которой принадлежит формула A в инструкциях assume и modify.

Синтаксис и семантика других ПО-выражений определяется аналогичным образом.

В заключение, определим функциональность основного модуля СППР СОМТИ — супервизора. Понятие супервизор имеет атрибуты обработчик-заданий, интерпретатор-конфигураций и адаптер, которые связывают соответствующие сущности КМ с супервизором. Например, присваивание ((обработчик-заданий is attribute of супервизор of обработчик-заданий) ::= true) связывает обработчик заданий с супервизором.

Функциональность понятия супервизор включает ПО-выражение (старт :: S) запуска супервизора S, задаваемое следующим правилом:

```
if (старт :: S) var S
```

```
then (assume (S is супервизор))
 ((получить-задание ::
   (адаптер of S)))
 (if ((value) is задание)
  then
   (обработать (value) ::
     (обработчик-заданий of S)
   (while (not ((value) is пустая
                конфигурация решения))
    do
     (выполнить (value) ::
       (интерпретатор-конфигурации of
         S))))
 (старт :: S)
```

Обработчик заданий получает на вход задание и возвращает конфигурацию решения, соответствующую этому заданию. Пустая конфигурация решения, определяемая ОП-выражением (_ is пустая конфигурация решения), соответствует решенной задаче. Функциональность понятия обработчик-заданий включает ПО-выражение (обработать A :: I), задаваемое правилом (if (обработать A :: I) var A I then (assume ((A is задание) and (I is обработчик-заданий))) (modify ((: value) is конфигурация-решения))). Заметим, что это правило расширяет функциональность обработчика заданий по сравнению с исходной КМ и, таким образом, оптимизирует ее, поскольку обработчик теперь может возвращать произвольную конфигурацию решения, а не только последовательность задач. Необходимость такой модификации КМ обоснована в [Анурееv, 2012].

Интерпретатор конфигурации получает на вход конфигурацию решения и возвращает модифицированную конфигурацию решения. Определим операционно-онтологическую семантику выражения (выполнить A :: I). Пусть ASSUME обозначает выражение (assume ((A is конфигурация-решения) and (I is интерпретатор-конфигурации))). Правило (if (выполнить A :: I) var A I then ASSUME (assume (not (A is задача)) and (not (A is модуль)))) (modify ((: value) is конфигурация-решения))) определяет выполнение конфигурации решения в случае, когда она не является задачей или модулем. Заметим, что в этом правиле не накладывается никаких ограничений на результат выполнения конфигурации решения кроме того, что он снова является конфигурацией решения. При конкретизации КМ посредством явного задания конструкторов конфигурации решения это правило разбивается на множество правил в зависимости от используемого конструктора. В случае, когда конфигурация решения является задачей или модулем, выполняются специализированные правила перехода, решающую конкретную задачу или выполняющие конкретный модуль.

4. Преимущества технологии и область ее применения

ФЛП-технология обеспечивает следующие преимущества при проектировании и прототипировании ИС:

- создается формальная логическая модель ИС. Более точно, модель базируется на комбинации операционного (помеченные системы переходов), онтологического (ОНТ-системы) и логического (алгебраические системы и логика безопасности) подходов;
- комбинируются различные подходы к обеспечению безопасности и оптимальности ИС (тестирование, экспертный анализ и дедуктивная верификация);
- формальная модель ИС приближена к описанию на естественном языке, что позволяет использовать ее в качестве документации ИС и при согласовании требований к ИС с заказчиком.

Область применимости ФЛП-технологии ограничивается ИС, которые описываются с помощью систем переходов с конечной структурой состояний (само множество состояний при этом может быть бесконечным) и функциональность которых может быть представлена на языке Atoment.

Заключение

В статье описана технология проектирования и прототипирования ИС с акцентом на их безопасность и концептуальную оптимальность.

Обычно под открытостью технологии поддержки того или иного этапа разработки ИС, понимается наличие либо грамотно спроектированного и документированного API к средствам разработки ИС, либо предметно-ориентированного языка разработки ИС (Этот язык может иметь различное представление: текстовое, графовое, в виде форм-шаблонов и т. д.). ФЛП-технология относится к открытым технологиям второго типа. Она не требует программирования. Проектирование и прототипирование ИС проводится экспертами в предметной области, для которой создается ИС, на языке выполнимых спецификаций Atoment в сотрудничестве с менеджерами проекта и разработчиками. Первые обеспечивают взаимодействие с заказчиком (в частности, согласование с ним формальной модели ИС и демонстрацию прототипа ИС). Вторые оценивают разрабатываемый прототип с точки зрения эффективности его реализации с помощью

конкретных программных средств и языков и на конкретных платформах.

Работа выполнена при финансовой поддержке интеграционного проекта РАН № 15/10 «Математические и методологические аспекты интеллектуальных информационных систем» и гранта РФФИ № 13-07-00457 «Методы и средства поддержки разработчиков систем поддержки принятия решений».

Библиографический список

[Анureев, 2009] Анureев И.С. Операционно-онтологический подход к формальной спецификации языков программирования // Программирование. 2009. № 1. С. 1–11.

[Анureев, 2012] Анureев И.С. Применение операционно-онтологического подхода к концептуальному моделированию систем поддержки принятия решений // Информационные и математические технологии в науке и управлении. Труды XVII Байкальской Всероссийской конференции. 2012. Том 3. С. 13–19.

[Анureев, 2008] Анureев И.С. Язык описания онтологических систем переходов OTSL как средство формальной спецификации программных систем // Вестник НГУ, серия Информационные технологии. 2008. Т. 6, № 3. С. 24–34.

[Голенков, 2011] Голенков В.В. Принципы построения массовой семантической технологии компонентного проектирования интеллектуальных систем // Материалы I международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем» (OSTIS-2011). Минск, 2011. С. 21–58.

[Грибова и др., 2012] Грибова В.В., Клещев А.С. Онтологическая парадигма программирования // Материалы II международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем» (OSTIS-2012). Минск, 2012. С. 213–220.

[Загорулько, 2009] Загорулько Ю.А. Технология разработки порталов научных знаний // Программные продукты и системы. 2009. № 4. С.25–29.

[Загорулько и др., 2010] Загорулько Ю.А., Анureев И.А., Загорулько Г.Б. Подход к разработке системы поддержки принятия решений на примере нефтегазодобывающего предприятия // Известия Томского политехнического университета. 2010. Т. 316, № 5. С. 127–131.

[Хорошевский, 2008] Хорошевский В.Ф. Пространства знаний в сети Интернет и Semantic Web (Часть 1) // Искусственный интеллект и принятие решений. 2008. № 1. С.80-97.

[Хорошевский, 2009] Хорошевский В.Ф. Пространства знаний в сети Интернет и Semantic Web (Часть 2) // Искусственный интеллект и принятие решений. 2008. № 4. С.15-36.

[Хорошевский, 2012а] Хорошевский, В.Ф. Пространства знаний в сети Интернет и Semantic Web (Часть 3) // Искусственный интеллект и принятие решений. 2012. № 1.

[Хорошевский, 2012б] Семантические технологии: ожидания и тренды // Материалы II международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем» (OSTIS-2012). Минск, 2012. С. 23–52.

[Anureev, 2012] Anureev I.S. Program specific transition systems // Joint NCC&IIS Bulletin, Series Computer Science. 2012. Vol. 33. P.1–21.

[Gurevich, 2004] Abstract State Machines: An Overview of the Project // Foundations of Information and Knowledge Systems (FoIKS): Proc. Third Internat. Symp. Lect. Notes Comput. Sci. 2004. Vol. 2942. P. 6–13.

OPEN TECHNOLOGY OF FORMAL LOGICAL DESIGN AND PROTOTYPING OF INTELLIGENT SYSTEMS

Anureev I.S., Atuchin M.M.

A.P. Ershov Institute of Informatics Systems Siberian Branch of the Russian Academy of Sciences, Novosibirsk, Russia

anureev@iis.nsk.su

atuchin.m@gmail.com

The paper presents a technology which allows one to choose an optimal conceptual configuration of an intelligent system at the stage of its design and to build its prototype on basis of formal logical methods. In particular, use of formal logical apparatus provides deductive verification of safety properties of the designed intelligent system. Use of the technology is illustrated by the example of specification of the conceptual model of a decision support system.

Introduction

The use of formal logical methods to provide the safety and efficiency of software becomes a standard for design and prototyping of industrial software systems (especially critical software). The resent trend to the use of intelligent components in such systems makes the challenge of developing formal methods for intelligent systems taking into account their specificity urgent.

It should also be noted that at present there is transition from the development of specific information systems to creation of technologies of their development. Therefore, an important task is to create technology based on formal logical methods (the FLP-technology) as applied to design and prototyping of information systems.

The paper proposes an approach to solve these two problems.

Main Part

The main part consists of four sections.

Section 1 describes the basis of the FLP-technology. This technology is based on specialized labeled transition systems aimed at accelerating the development of the specification and verification of software systems and the executable specification language Atoment used to specify these systems.

Section 2 describes the stages of the design and prototyping of information systems based on the FLP technology.

Section 3 illustrates the use of FLP-technology by the example of specification of the conceptual model of a decision support system.

Section 4 describes the advantages of FLP technology and defines its application domain.

Conclusion

The paper describes the technology of design and prototyping of information systems, with emphasis on their safety and conceptual-optimality properties.