# VDMTools®

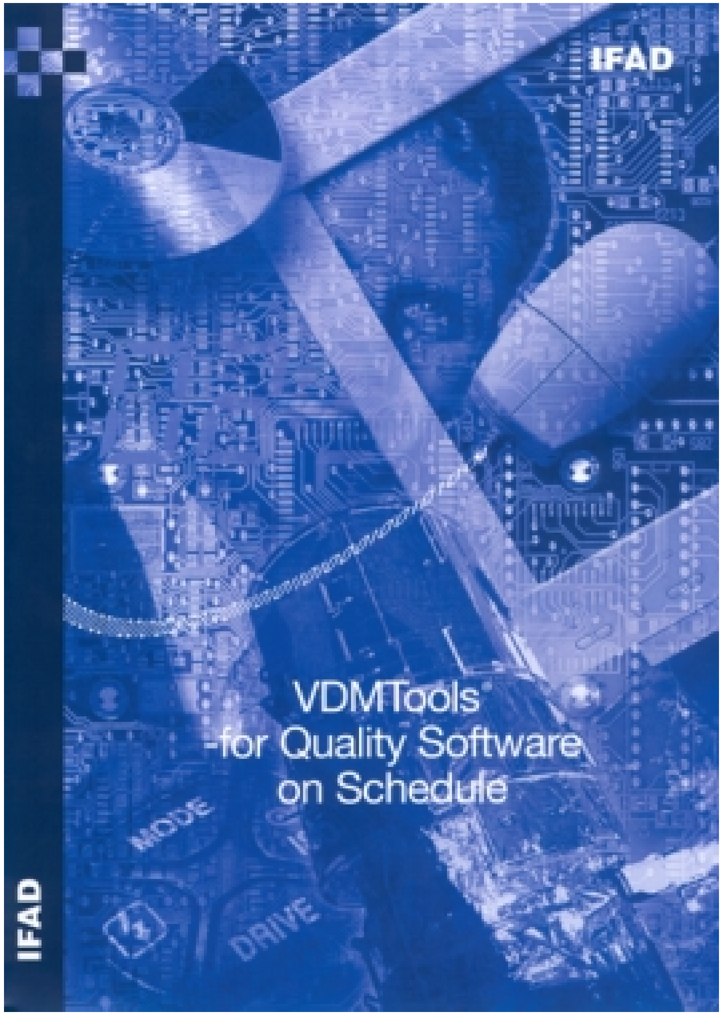
## Validated Design through Modelling

# VDM Tool Support

www.ifad.dk

**IFAD A/S**
Forskerparken 10
DK-5230 Odense M
Denmark

# VDMTools



IFAD

VDMTools®
for Quality Software
on Schedule

IFAD

TECHNOLOGY

## Easier Development of Better Software

**Flexible Integration**
VDMTools® offers a wide collection of advanced analysis and debugging tools to aid in software development. The central idea is to automate as much as possible of the usually manual checking of design documents in order to avoid the common problems of substantial redesign at a late stage. VDMTools® allows flexible integration with almost any development environment and provides specific support for visual modeling through round trip engineering with market leading UML tools, including Rational Rose.

**Precise Modeling**
Most of the quality problems in software development arise due to the use of ad hoc methods in the early stages. Usually these are based on notations such as natural language and diagrams, which support only manual inspections. Therefore these methods are expensive to use and less effective in locating errors.

VDMTools® supports precise modeling in structured and well-defined notations: the ISO Standard specification language VDM-SL and its object-oriented extension VDM++. These enable automatic analysis to ensure that models are consistent and complete. Moreover, the VDM notations support developers in focusing on high-level conceptual questions concerning data and behavioral aspects of the system rather than low-level implementation details. Software development becomes productive and intriguing!

**Automatic Checking**
VDMTools® provides powerful and easy-to-use facilities for systematic analysis based on standard software engineering practice such as testing. The core technology is an advanced interpreter that performs automatic consistency checking during execution of a model. Testing can be conducted both in an interactive fashion and by running a test environment. VDMTools® also provides a powerful interactive debugger that offers the usual functionality at the model level such as breakpoints and stepping.

Through its unique support for automatic checking, VDMTools® goes way beyond traditional methods in allowing developers to get direct feedback on their understanding of a system. VDMTools® acknowledges the importance of not only being able to describe one's understanding in a model, but also being able to demonstrate this understanding, even in front of a client who has no prior experience with VDMTools®.

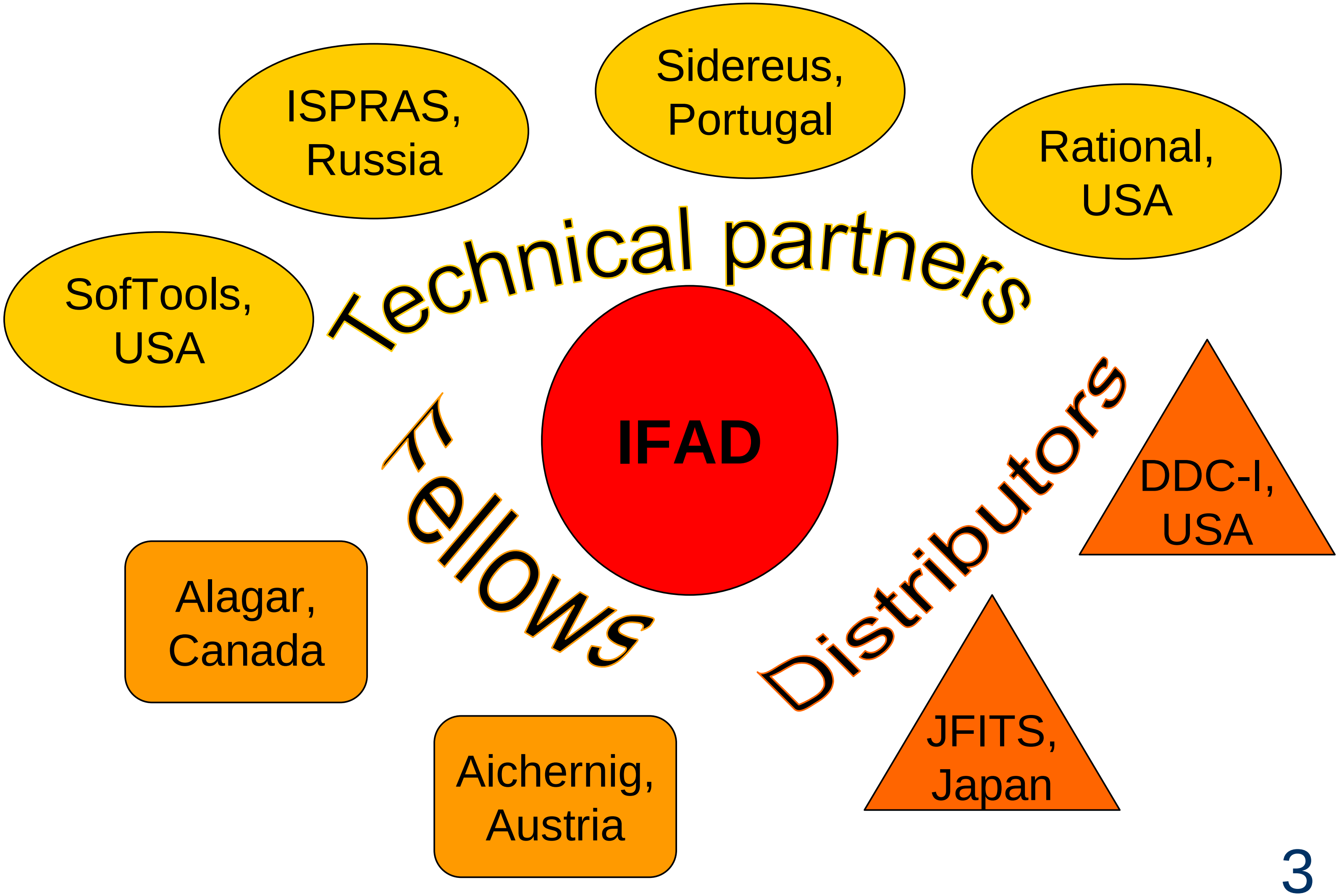
Additional features are available to support such "rapid prototyping". These include a dynamic link facility and a Corba compliant API facility, which allow any piece of code such as a graphical front-end or existing legacy code to be executed together with a VDM model. Through these facilities, VDMTools® enables invaluable feedback to be obtained from a client early in the development, before expensive commitment is made to an implementation.

**Code Generation**
VDMTools® supports automatic code generation from design models to implementation languages, including C++. This eliminates the tedious coding work in software development and makes it easy to keep documentation and code in conformance with each other. The tools generate fully executable code, so most models can be translated without human intervention if desired. The tools provide facilities for including user-defined code for non-executable parts of a model.

THE IFAD® VDMTools

**Technical Information**
VDMTools® includes the following products: The IFAD VDM-SL Toolbox, The IFAD VDM++ Toolbox, The Rose-VDM++ Link (for Rational Rose), Dynamic Link Facility, Corba Compliant API, Code Generation, and Document Generation (Microsoft Word). VDMTools® runs on Windows and a number of Unix platforms.

# IFAD VDMTools Alliances



IFAD

Technical partners

ISPRAS, Russia

Sidereus, Portugal

Rational, USA

SofTools, USA

Fellows

Alagar, Canada

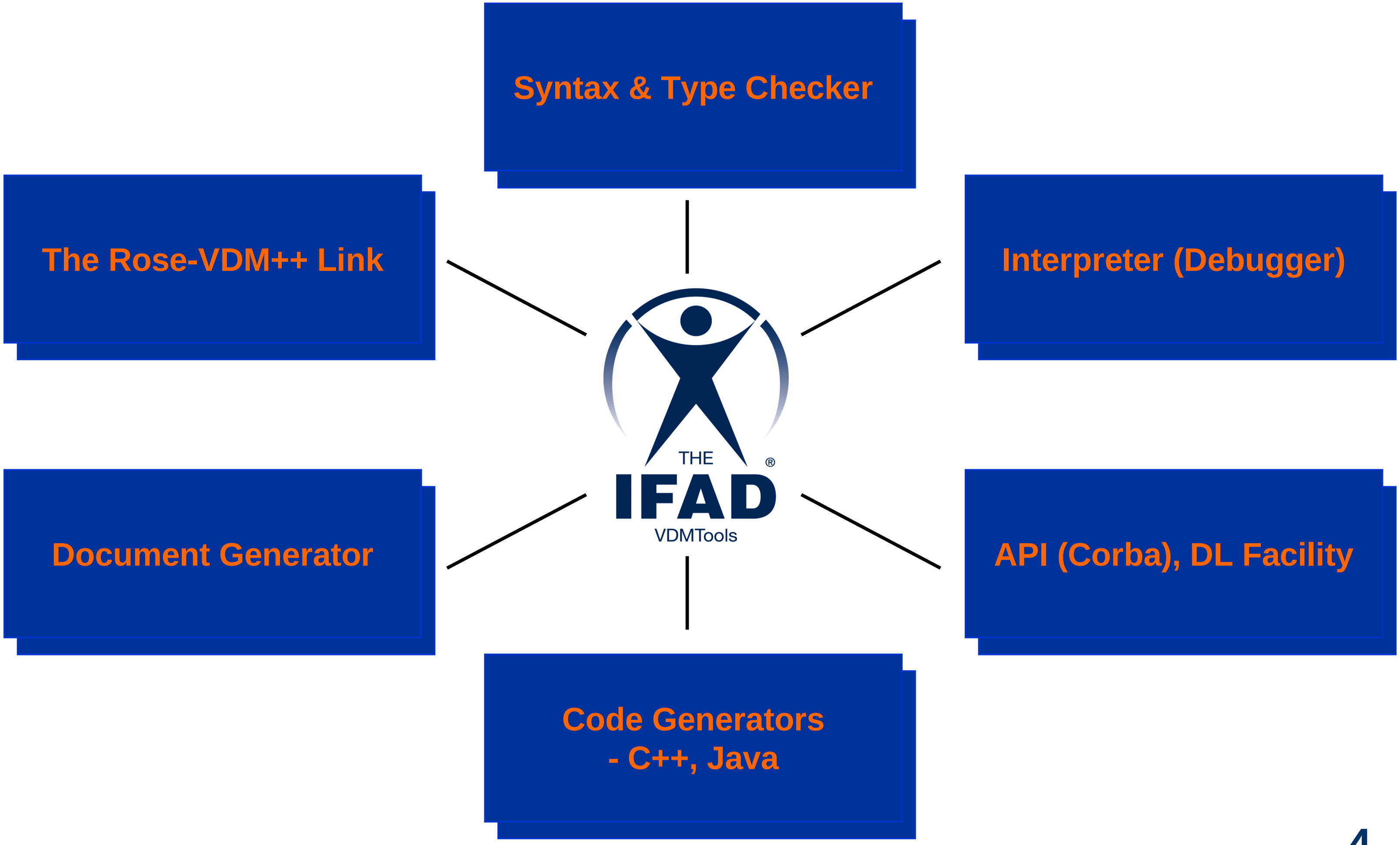
Aichernig, Austria

Distributors

DDC-I, USA

JFITS, Japan

# VDMTools® Overview



Syntax & Type Checker

The Rose-VDM++ Link

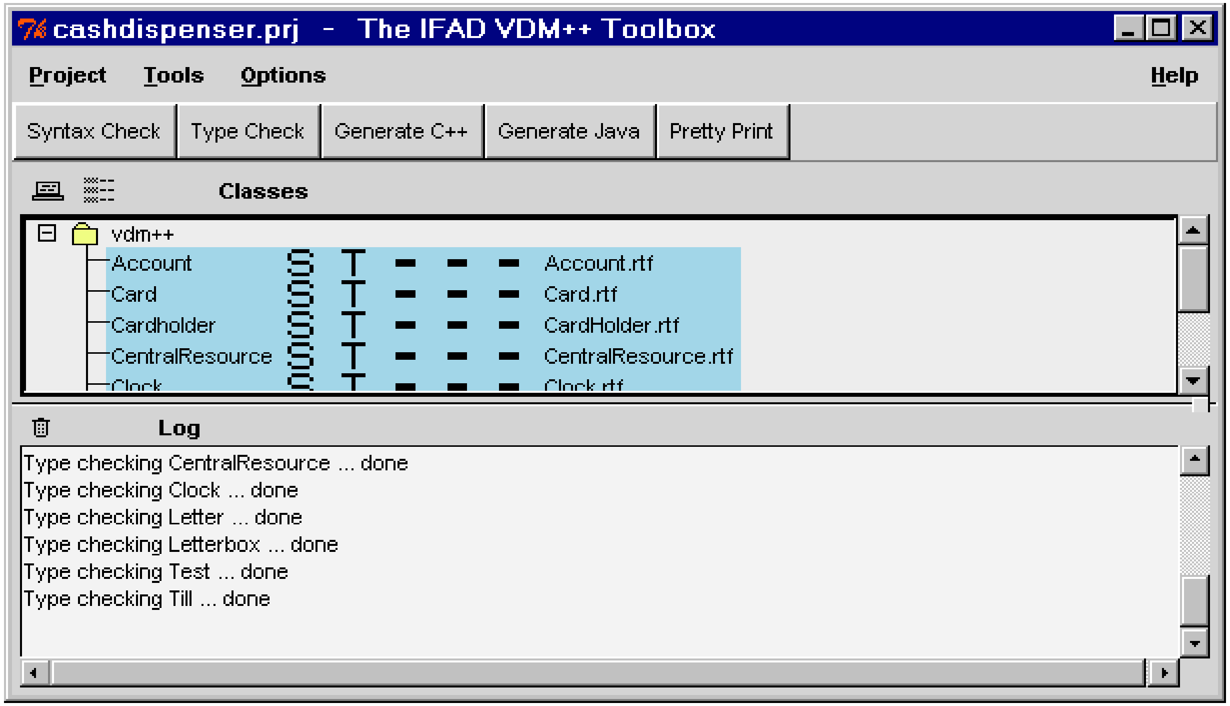
Interpreter (Debugger)

Document Generator

API (Corba), DL Facility

Code Generators
- C++, Java

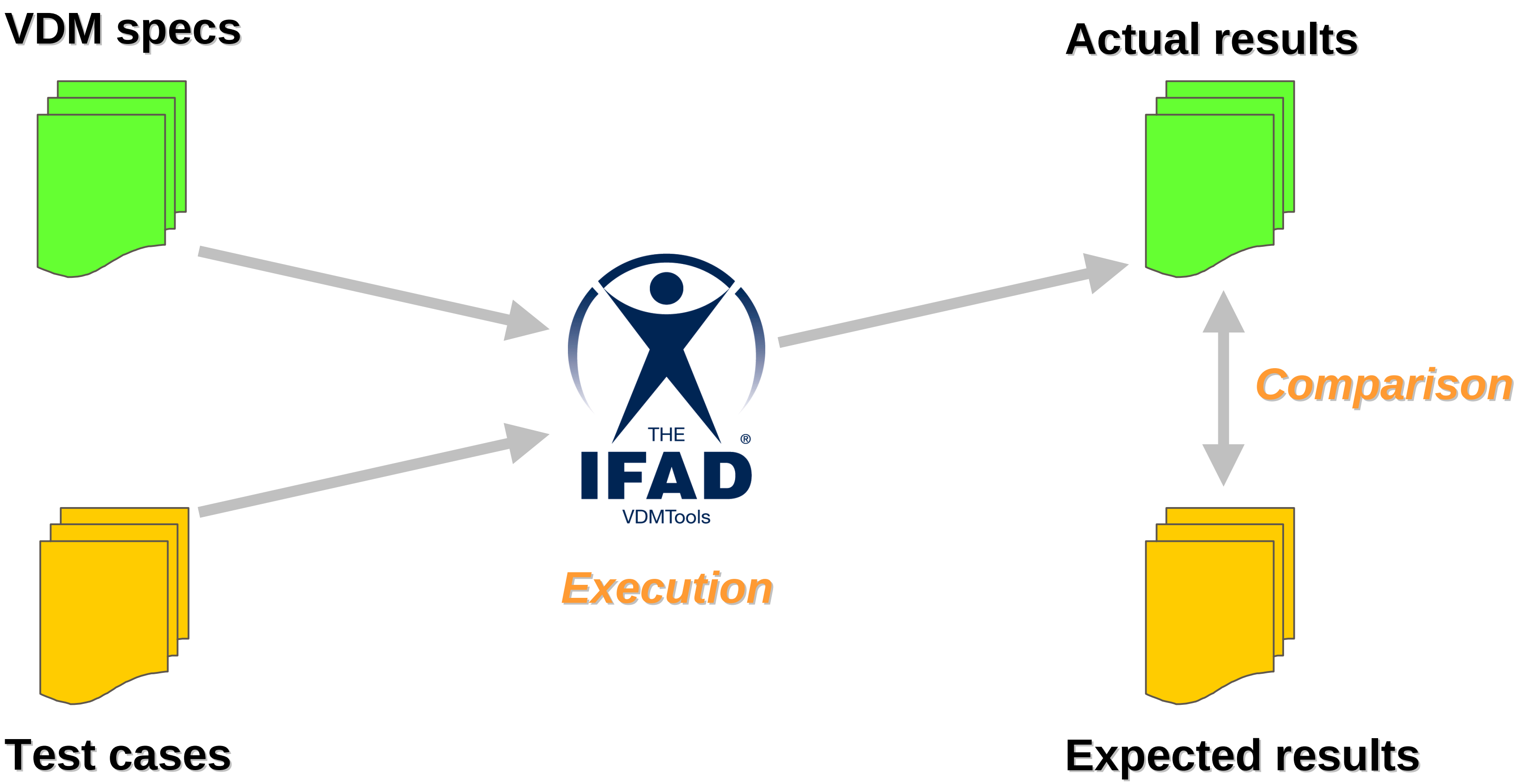
# Syntax and Type Checking



| vdm++ | | | | | | |
|---|---|---|---|---|---|---|
| Account | S | T | ▬ | ▬ | ▬ | Account.rtf |
| Card | S | T | ▬ | ▬ | ▬ | Card.rtf |
| Cardholder | S | T | ▬ | ▬ | ▬ | CardHolder.rtf |
| CentralResource | S | T | ▬ | ▬ | ▬ | CentralResource.rtf |
| Clock | S | T | ▬ | ▬ | ▬ | Clock.rtf |

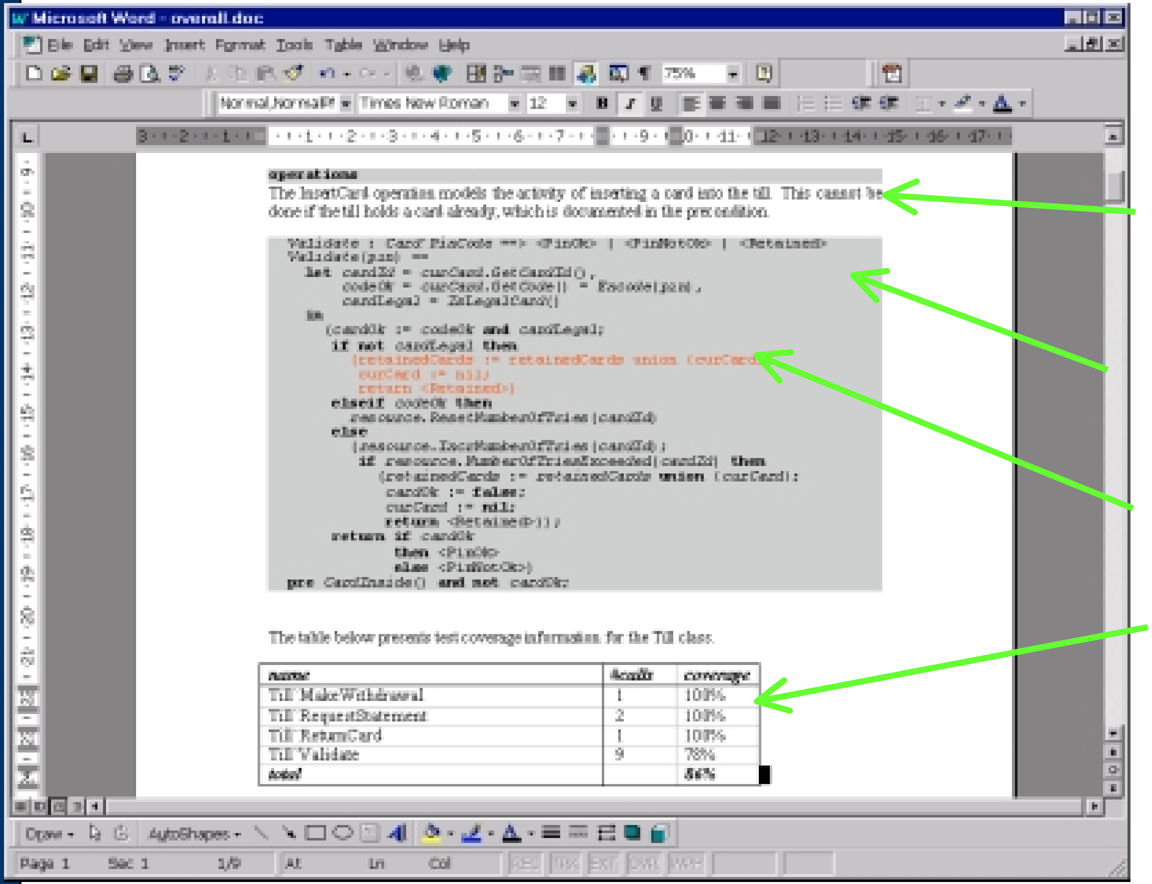
```
Type checking CentralResource ... done
Type checking Clock ... done
Type checking Letter ... done
Type checking Letterbox ... done
Type checking Test ... done
Type checking Till ... done
```

# Validation with VDMTools®



# Documentation in MS Word/RTF

One compound document:



**operations**

The InsertCard operation models the activity of inserting a card into the till. This cannot be done if the till holds a card already, which is documented in the precondition.

```
  Validate : Card`PinCode ==> <PinOk> | <PinNotOk> | <Retained>
  Validate(pin) ==
    let cardId = curCard.GetCardId(),
        codeOk = curCard.GetCode() = Encode(pin),
        cardLegal = IsLegalCard()
    in
      (cardOk := codeOk and cardLegal;
       if not cardLegal then
         (retainedCards := retainedCards union {curCard
          curCard := nil;
          return <Retained>)
       elseif codeOk then
         resource.ResetNumberOfTries(cardId)
       else
         (resource.IncrNumberOfTries(cardId);
          if resource.NumberOfTriesExceeded(cardId) then
            (retainedCards := retainedCards union {curCard};
             cardOk := false;
             curCard := nil;
             return <Retained>));
       return if cardOk
              then <PinOk>
              else <PinNotOk>)
  pre CardInside() and not cardOk;
```

The table below presents test coverage information for the Till class.

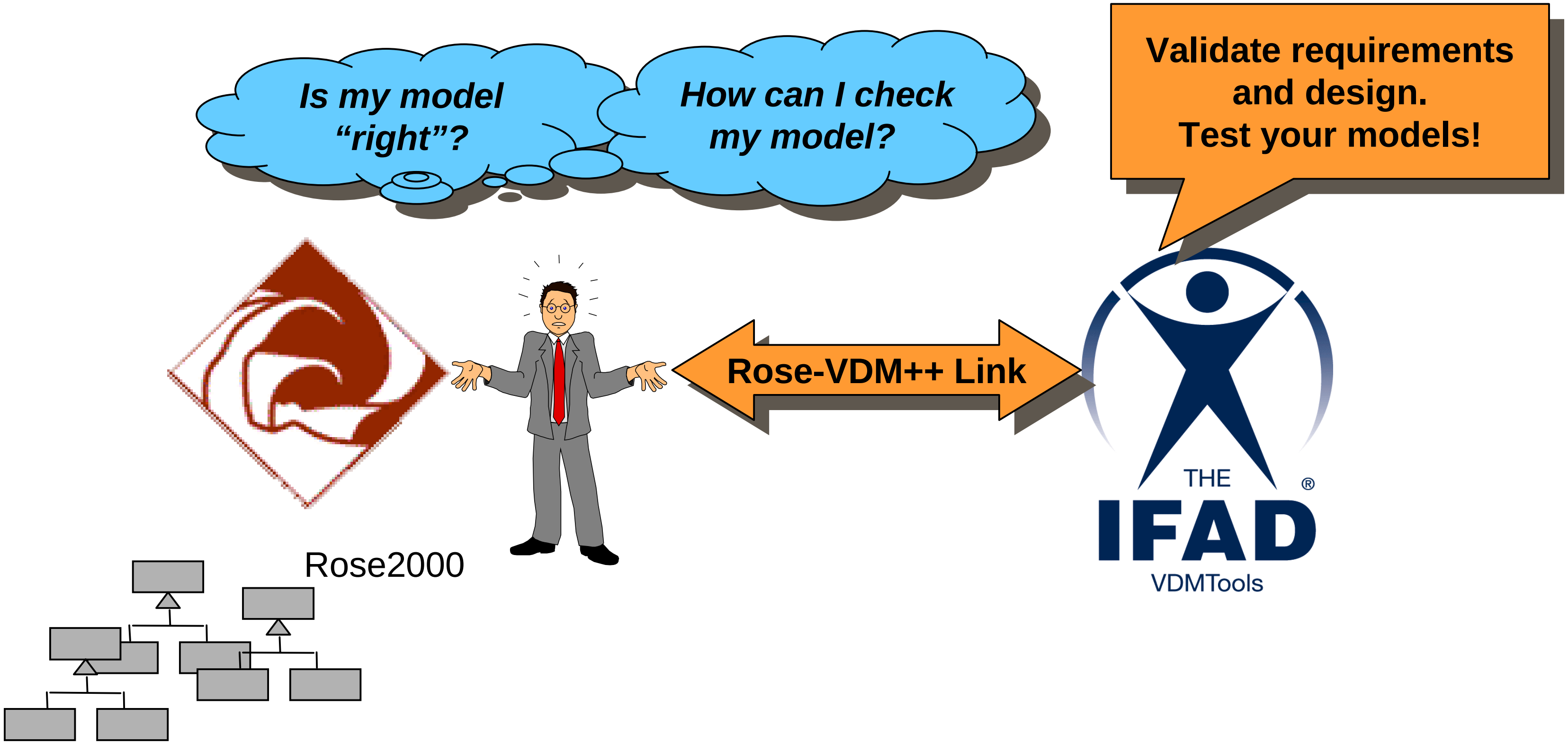
| *name* | *#calls* | *coverage* |
|---|---|---|
| Till`MakeWithdrawal | 1 | 100% |
| Till`RequestStatement | 2 | 100% |
| Till`ReturnCard | 1 | 100% |
| Till`Validate | 9 | 78% |
| *total* | | *86%* |

- Documentation
- Specification
- Test coverage
- Test coverage statistics

# The Rose-VDM++ Link

- Supports round-trip engineering with Rational Rose
- Offers the complementary benefits of the graphical notation UML and the textual formal notation VDM++
- Massive use of UML expected world-wide!

# The Rose-VDM++ Link



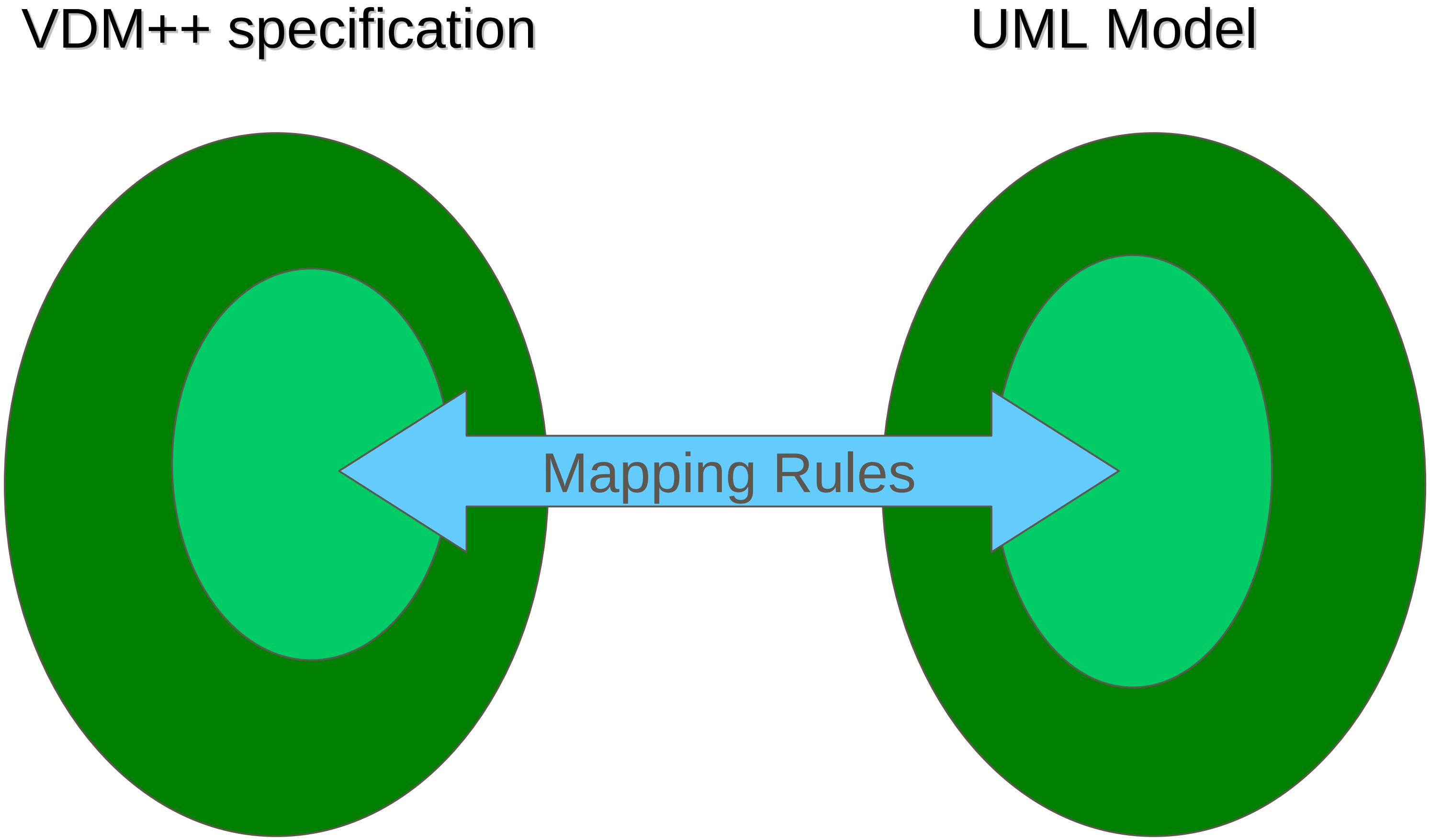
*Is my model "right"?*

*How can I check my model?*

**Validate requirements and design. Test your models!**

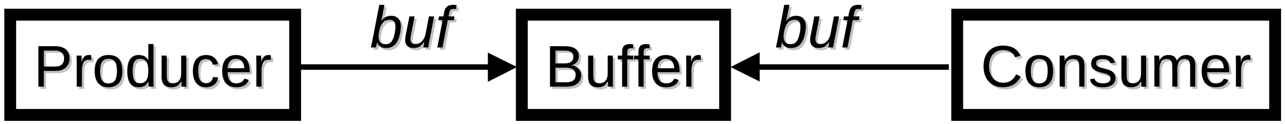
**Rose-VDM++ Link**

# Integration Principle



# Associations

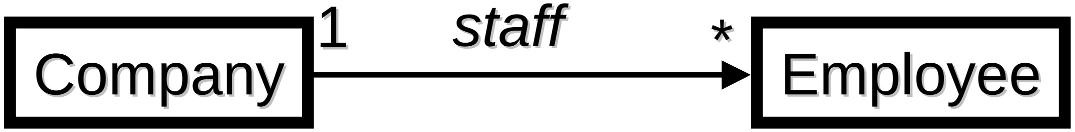
- Clientship relations are represented in UML as an association:



```
class Producer
instance variables
 buf: Buffer
 ...
```

```
class Consumer
instance variables
  buf: Buffer
  ...
```

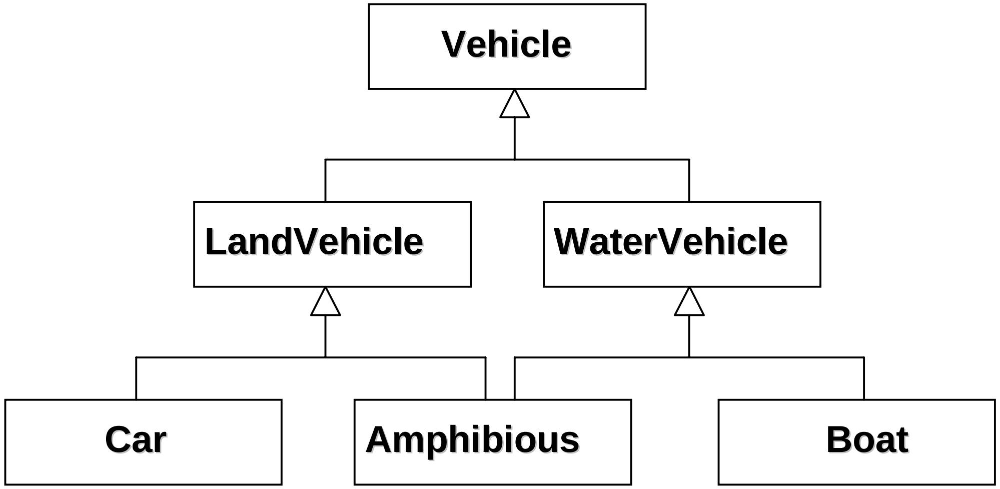
- Associations can have multiplicity



```
class Company
 instance variables
  staff: set of Employee;
  ...
```

# Inheritance

- In UML inheritance is termed as *generalization*.



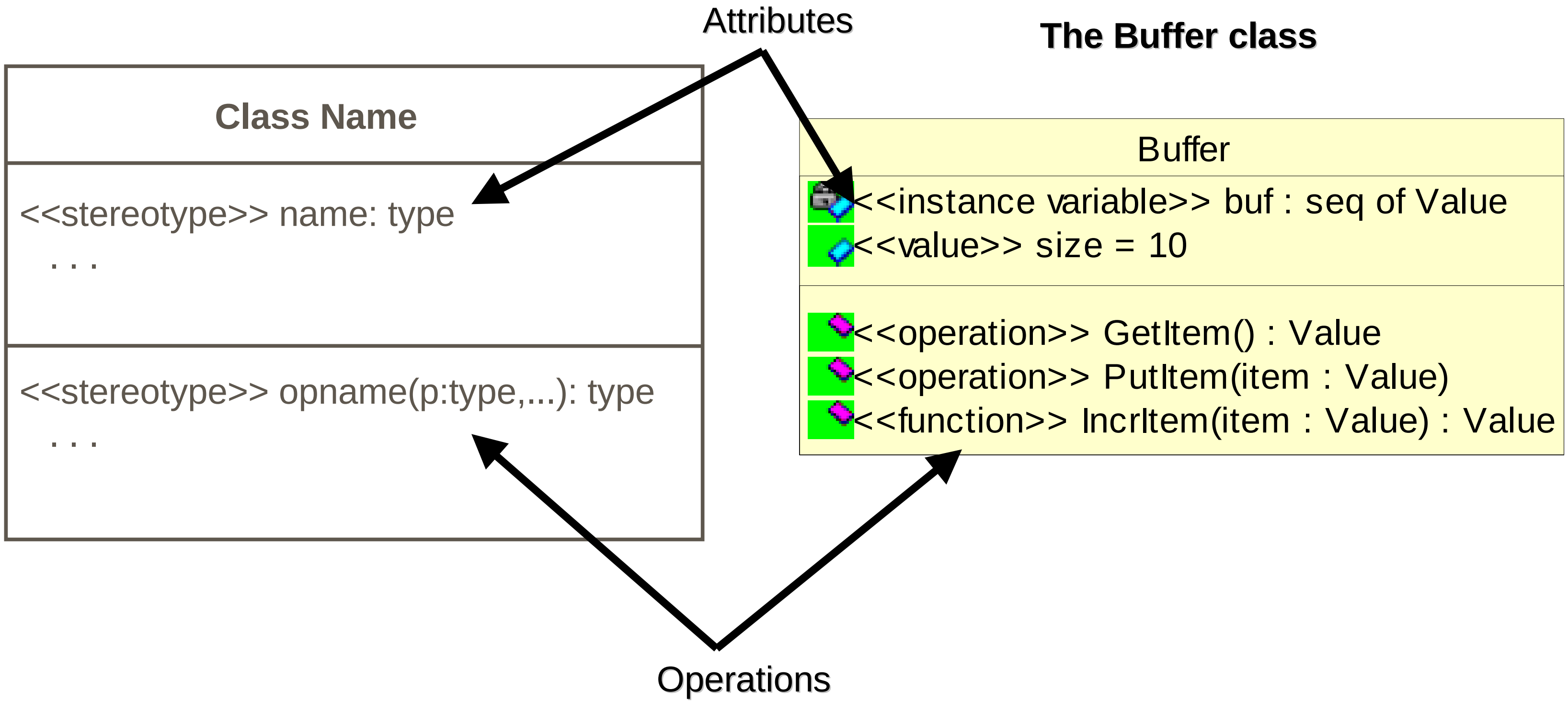
- In VDM++ the “is subclass of” keyword identifies the inheritance relations between classes

```
class Vehicle
 ...
end Vehicle
```

```
class LandVehicle
 is subclass of Vehicle
 ...
end LandVehicle
```

```
class Amphibious
 is subclass of
    LandVehicle,
    WaterVehicle
 ...
end Amphibious
```

# A Class Diagram



Attributes

The Buffer class

Class Name

<<stereotype>> name: type
. . .

<<stereotype>> opname(p:type,...): type
. . .

Buffer

<<instance variable>> buf : seq of Value
<<value>> size = 10

<<operation>> GetItem() : Value
<<operation>> PutItem(item : Value)
<<function>> IncrItem(item : Value) : Value

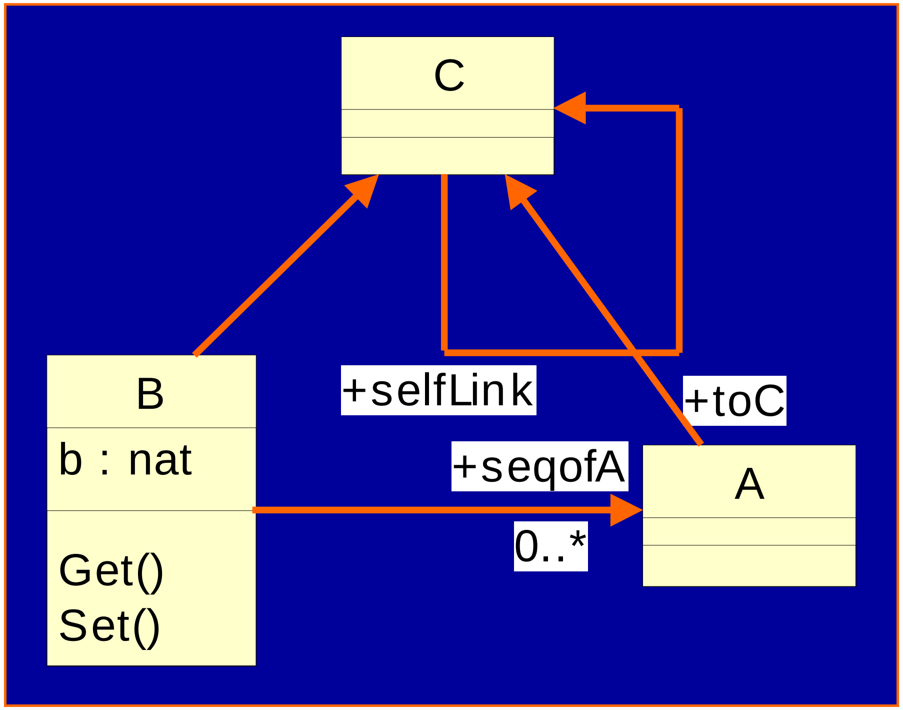
Operations

# Mapping Rules

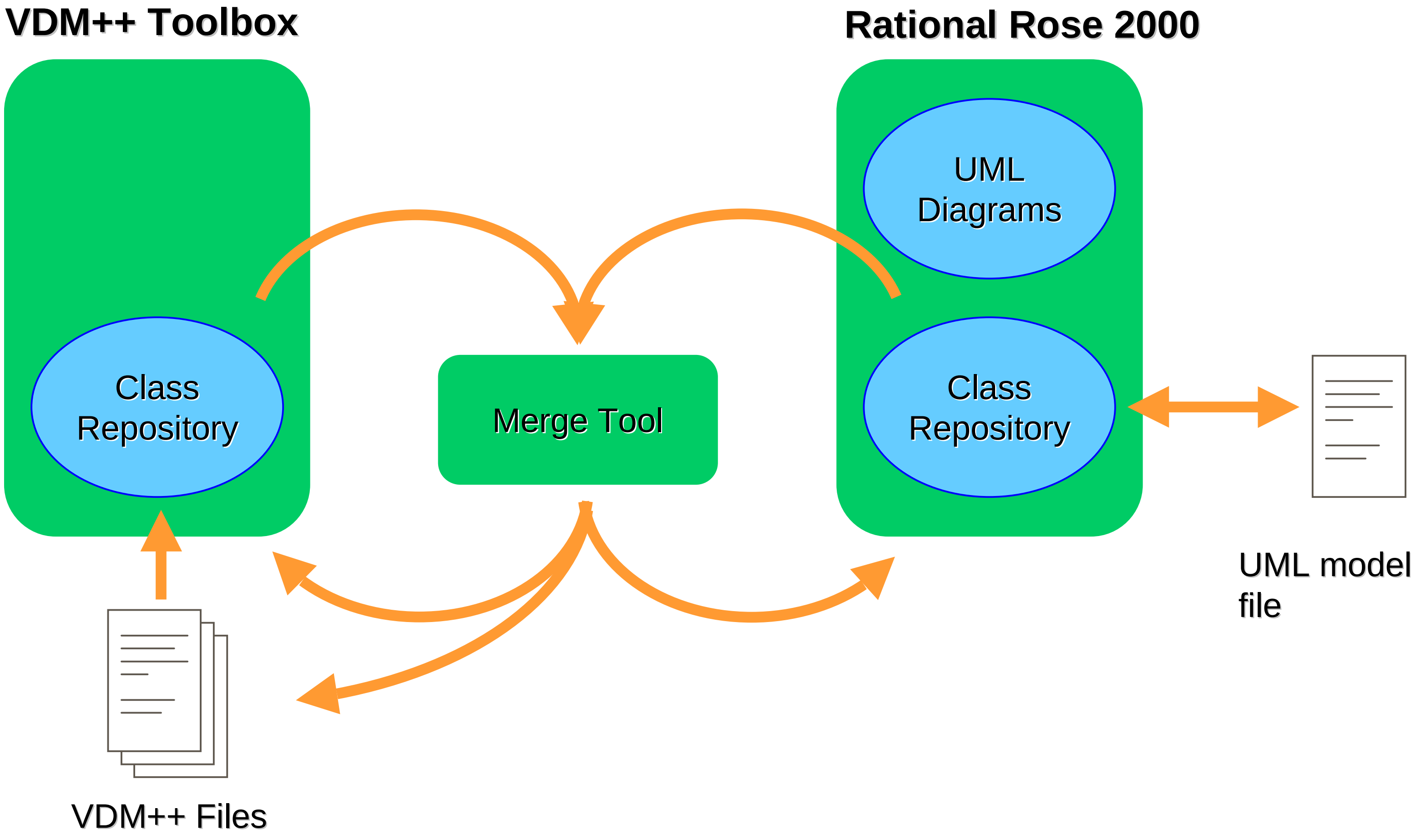
```
class A
  instance variables
    toC: C
end A
```

```
class B is subclass of C
  instance variables
    b: nat;
    seqofA: seq of A
  operations
   public Get: () ==> nat
   Get() ==
     return b;
   public Set: nat ==> ()
   Set(val) ==
     b := val
end B
```

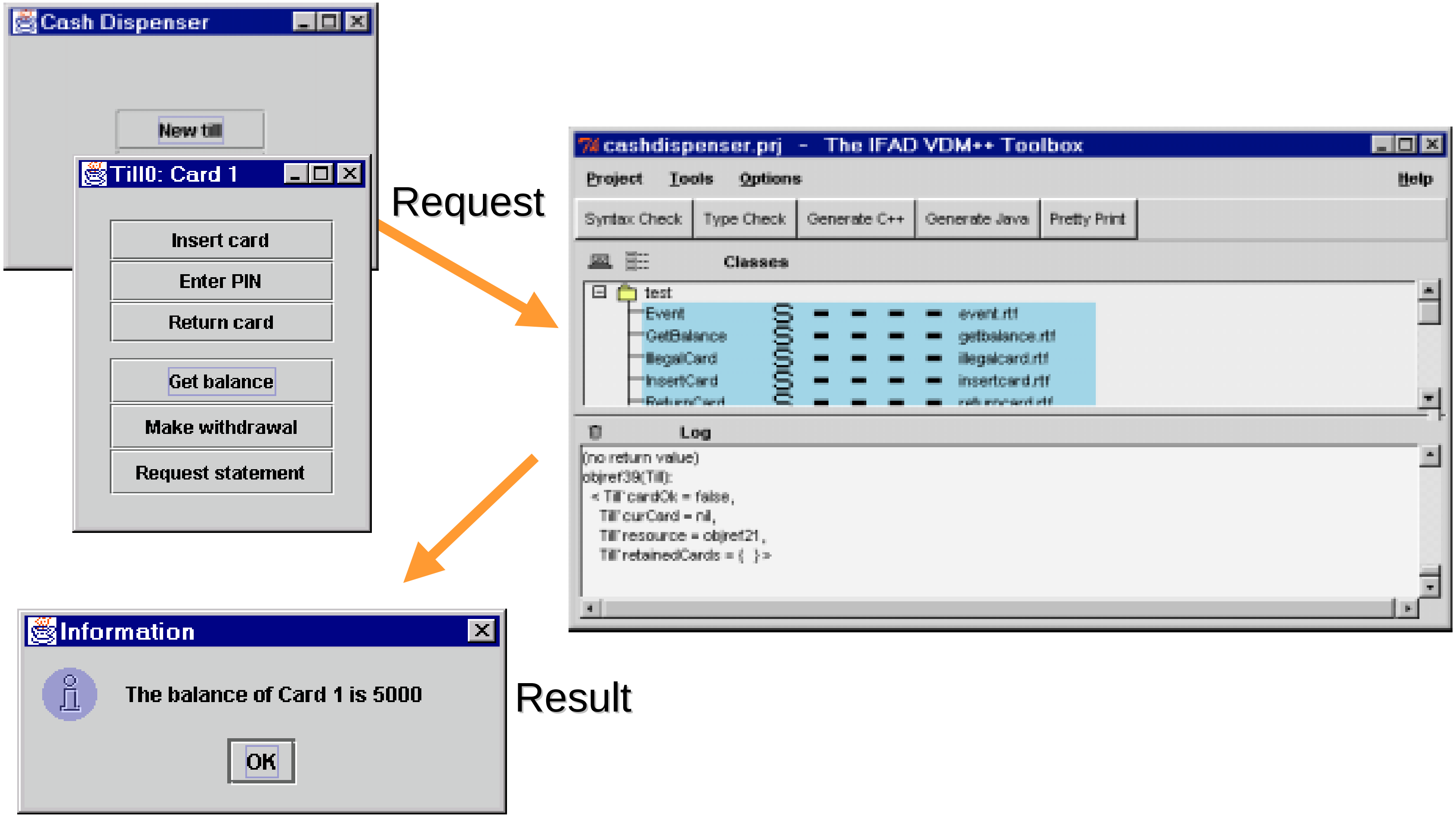
```
class C
  instance variables
    selfLink: C
end C
```



# Architecture of Link



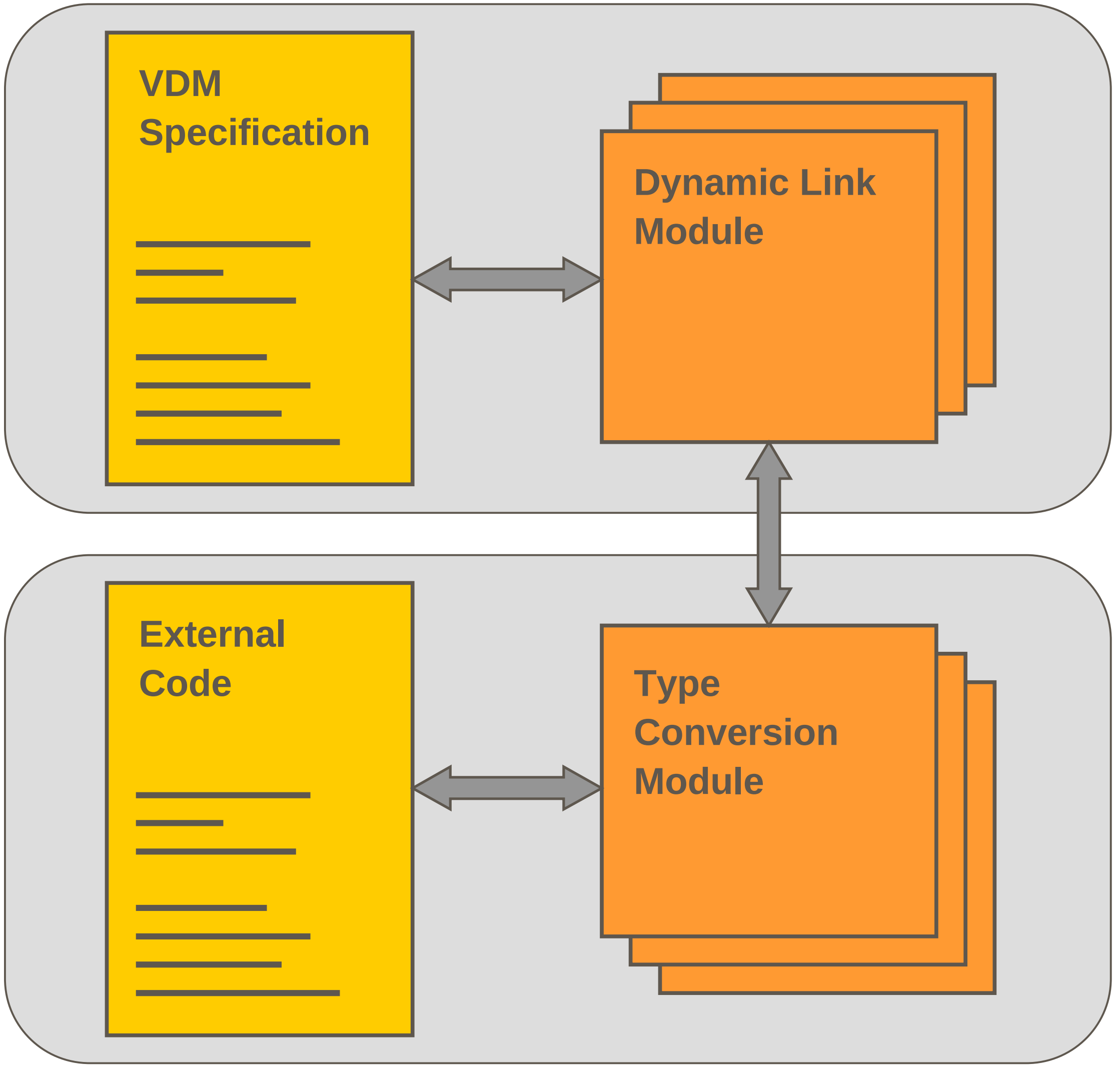
# Toolbox API



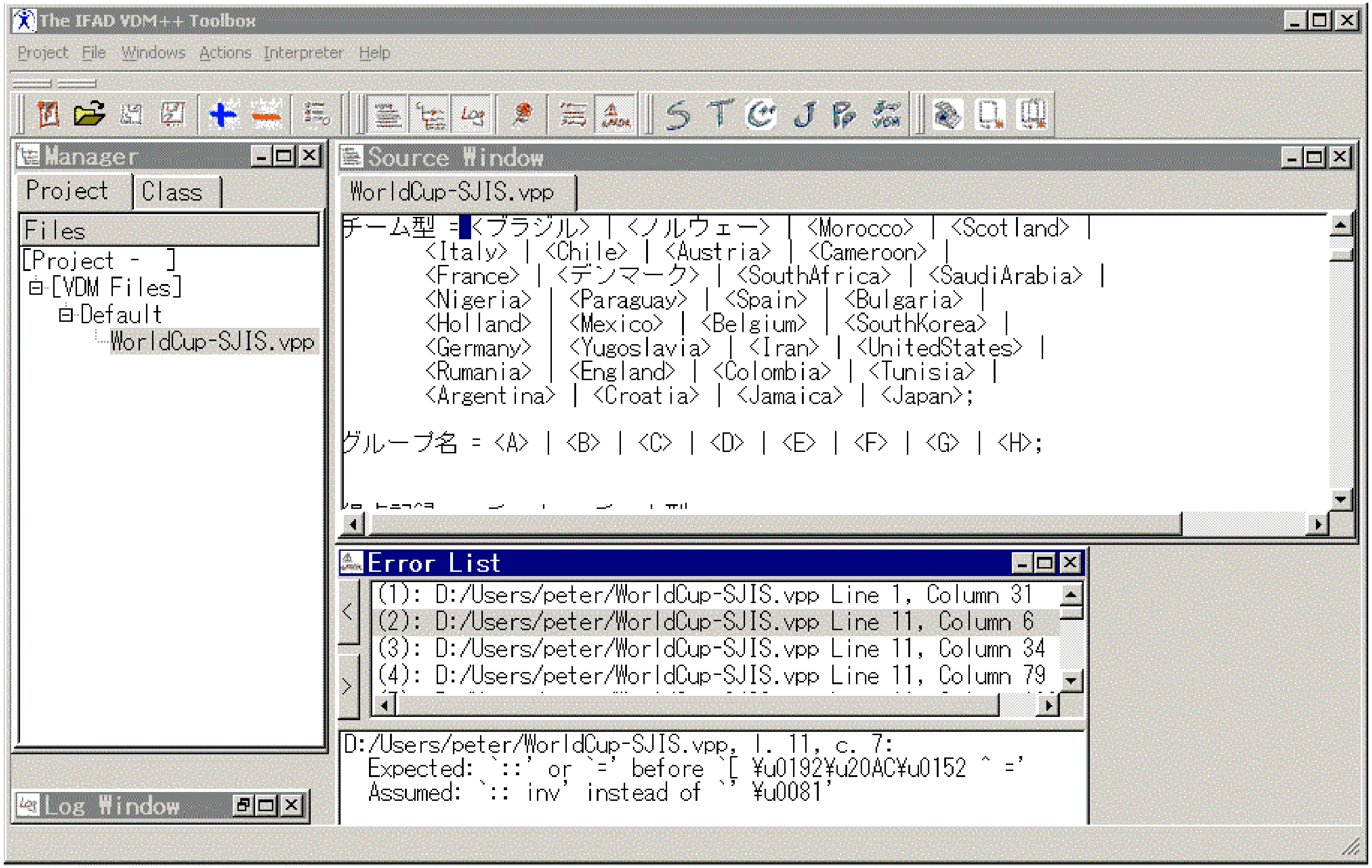
```
(no return value)
objref39(Till):
 < Till`cardOk = false,
   Till`curCard = nil,
   Till`resource = objref21,
   Till`retainedCards = {  }>
```

The balance of Card 1 is 5000

# Dynamic Link Facility



# Japanese Support



WorldCup-SJIS.vpp

```
チーム型 =<ブラジル> | <ノルウェー> | <Morocco> | <Scotland> |
        <Italy> | <Chile> | <Austria> | <Cameroon> |
        <France> | <デンマーク> | <SouthAfrica> | <SaudiArabia> |
        <Nigeria> | <Paraguay> | <Spain> | <Bulgaria> |
        <Holland> | <Mexico> | <Belgium> | <SouthKorea> |
        <Germany> | <Yugoslavia> | <Iran> | <UnitedStates> |
        <Rumania> | <England> | <Colombia> | <Tunisia> |
        <Argentina> | <Croatia> | <Jamaica> | <Japan>;

グループ名 = <A> | <B> | <C> | <D> | <E> | <F> | <G> | <H>;
```

```
(1): D:/Users/peter/WorldCup-SJIS.vpp Line 1, Column 31
(2): D:/Users/peter/WorldCup-SJIS.vpp Line 11, Column 6
(3): D:/Users/peter/WorldCup-SJIS.vpp Line 11, Column 34
(4): D:/Users/peter/WorldCup-SJIS.vpp Line 11, Column 79
```

```
D:/Users/peter/WorldCup-SJIS.vpp, l. 11, c. 7:
  Expected: `::' or `=' before `[ ¥u0192¥u20AC¥u0152 ^ ='
  Assumed: `:: inv' instead of `' ¥u0081'
```

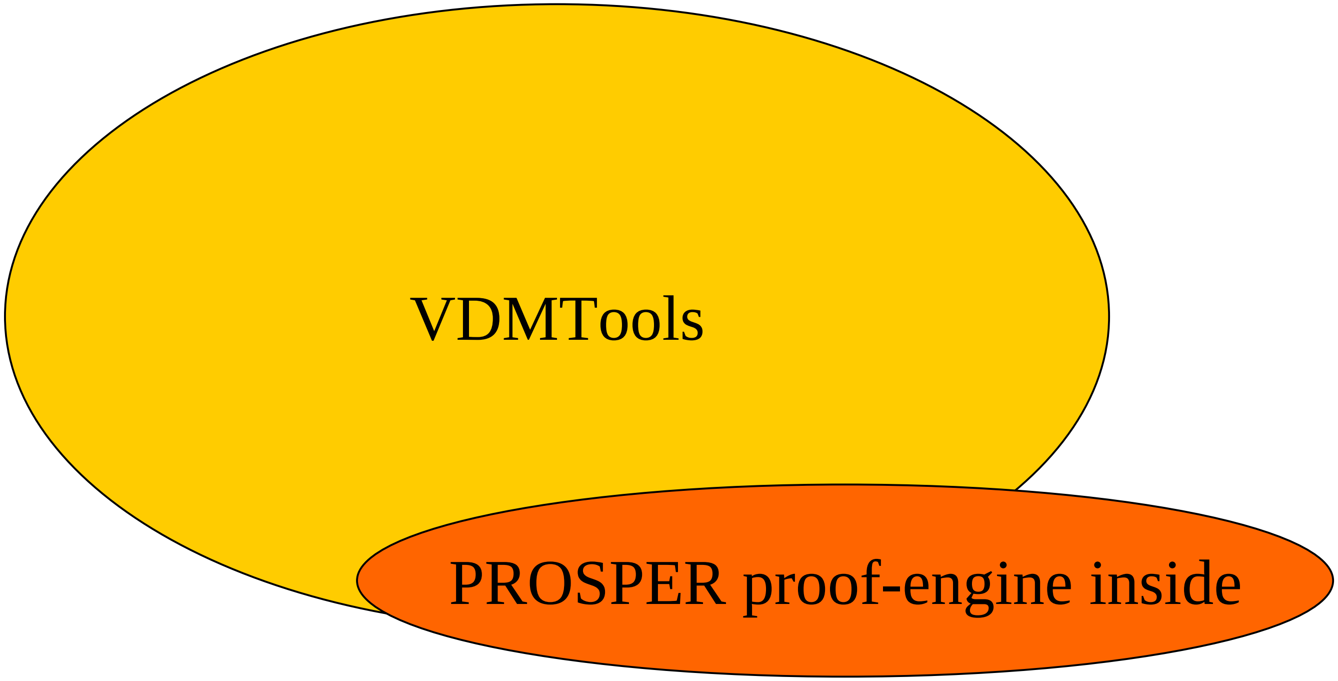
# Free Academic Site Licenses

- For teaching purposes
- For research purposes
- So far more than 30 around the world
- Fitzgerald&Larsen book translated to Japanese
- A VDM++ book to be published 2002

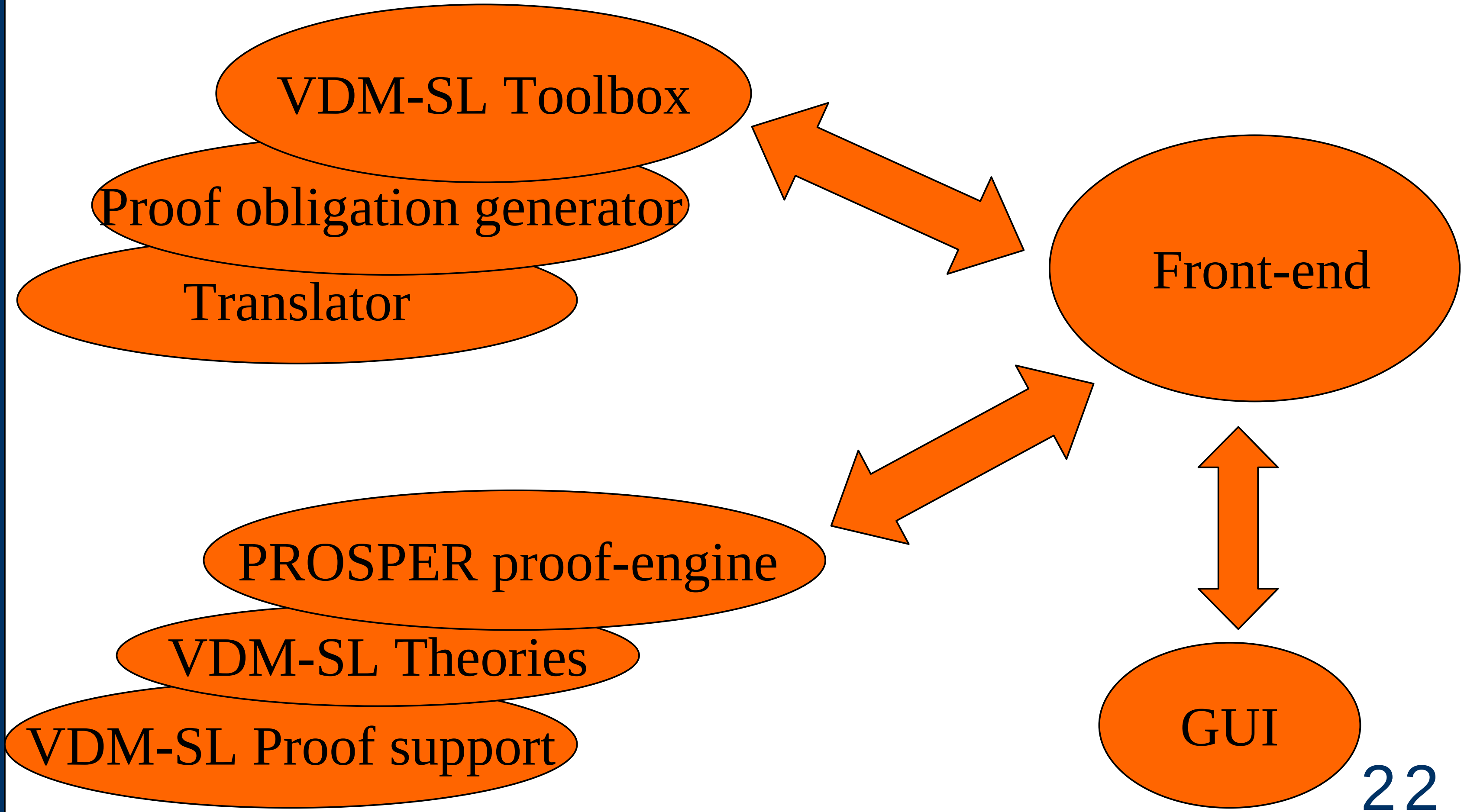
# Future VDMTools Extensions

- Reverse engineering from Java
- Real-time features
- Proof support
- Test case generation (ISPRAS)
- Database reverse engineering (Sidereus)
- No more EU projects
- Directions will depend on customers

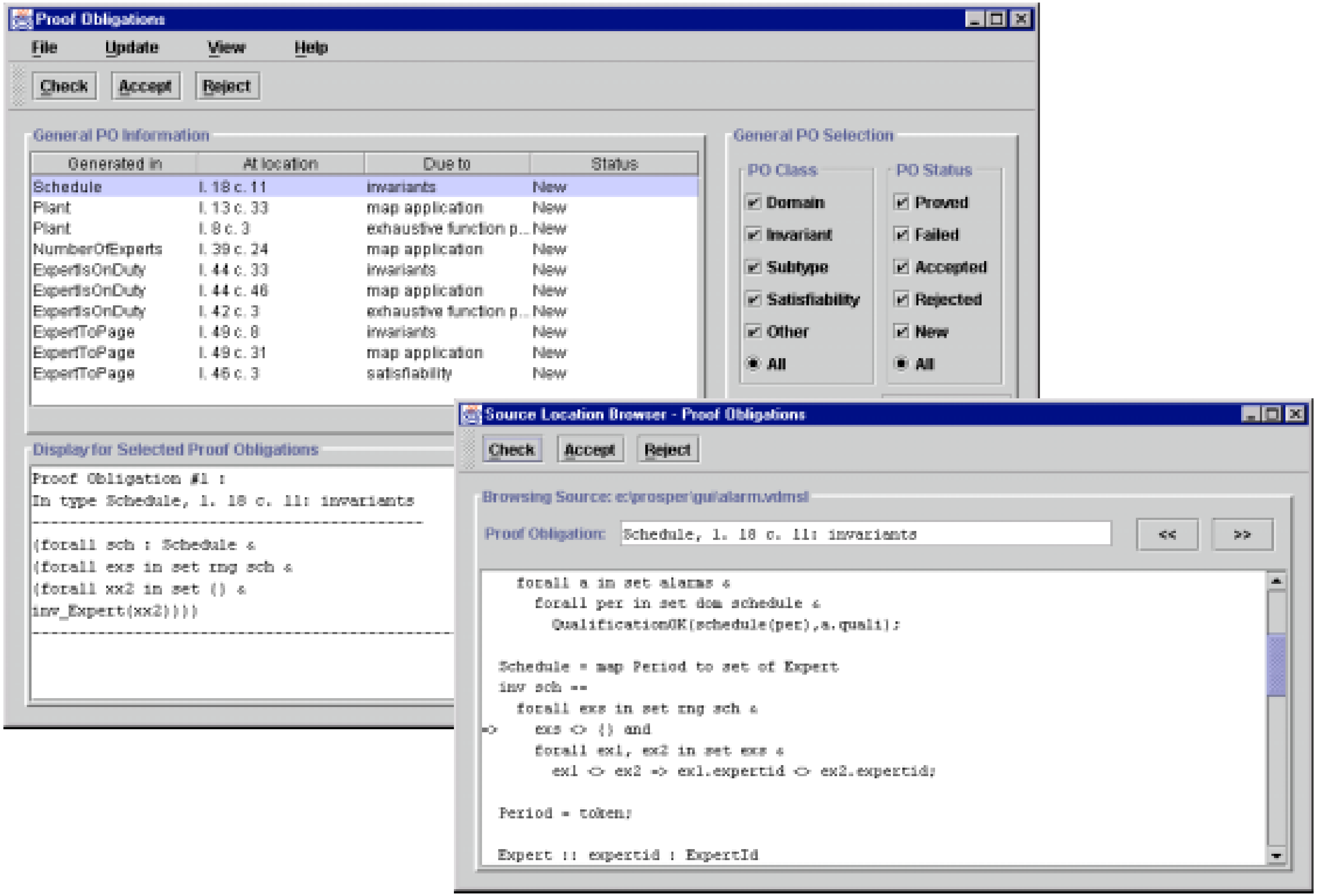
# New Proof-support extensions



# PROSPER Component View



# Proof obligation generator



General PO Information

| Generated in | At location | Due to | Status |
|---|---|---|---|
| Schedule | l. 18 c. 11 | invariants | New |
| Plant | l. 13 c. 33 | map application | New |
| Plant | l. 8 c. 3 | exhaustive function p... | New |
| NumberOfExperts | l. 39 c. 24 | map application | New |
| ExpertIsOnDuty | l. 44 c. 33 | invariants | New |
| ExpertIsOnDuty | l. 44 c. 46 | map application | New |
| ExpertIsOnDuty | l. 42 c. 3 | exhaustive function p... | New |
| ExpertToPage | l. 49 c. 8 | invariants | New |
| ExpertToPage | l. 49 c. 31 | map application | New |
| ExpertToPage | l. 46 c. 3 | satisfiability | New |

General PO Selection

PO Class
- ☑ Domain
- ☑ Invariant
- ☑ Subtype
- ☑ Satisfiability
- ☑ Other
- ● All

PO Status
- ☑ Proved
- ☑ Failed
- ☑ Accepted
- ☑ Rejected
- ☑ New
- ● All

Display for Selected Proof Obligations

```
Proof Obligation #1 :
In type Schedule, l. 18 c. 11: invariants
----------------------------------------------
(forall sch : Schedule &
(forall exs in set rng sch &
(forall xx2 in set {} &
inv_Expert(xx2))))
----------------------------------------------
```

Browsing Source: e:\prosper\gui\alarm.vdmsl

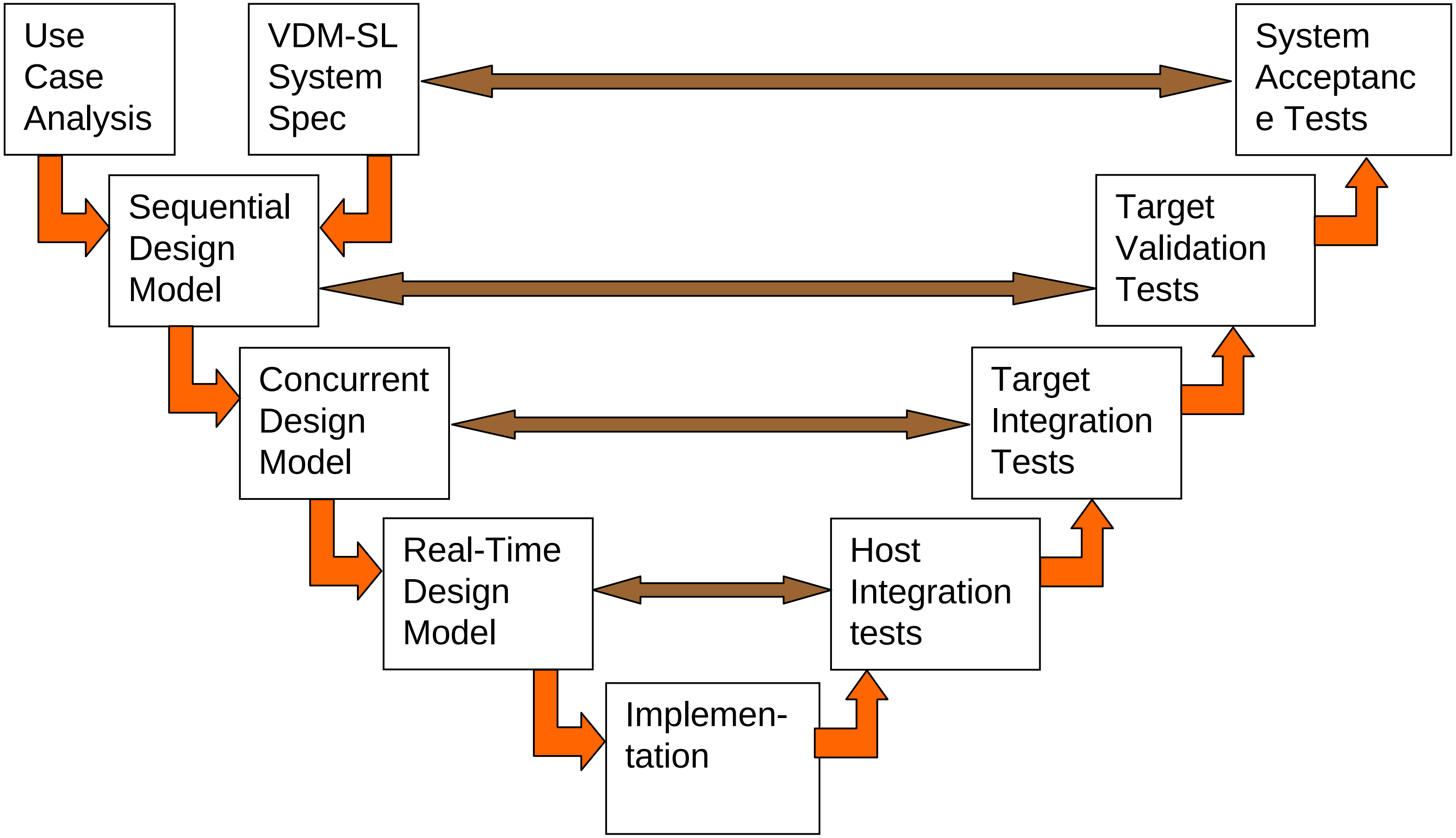
Proof Obligation: Schedule, l. 18 c. 11: invariants

```
    forall a in set alarms &
      forall per in set dom schedule &
        QualificationOK(schedule(per),a.quali);

  Schedule = map Period to set of Expert
  inv sch ==
    forall exs in set rng sch &
=>    exs <> {} and
      forall ex1, ex2 in set exs &
        ex1 <> ex2 => ex1.expertid <> ex2.expertid;

  Period = token;

  Expert :: expertid : ExpertId
```

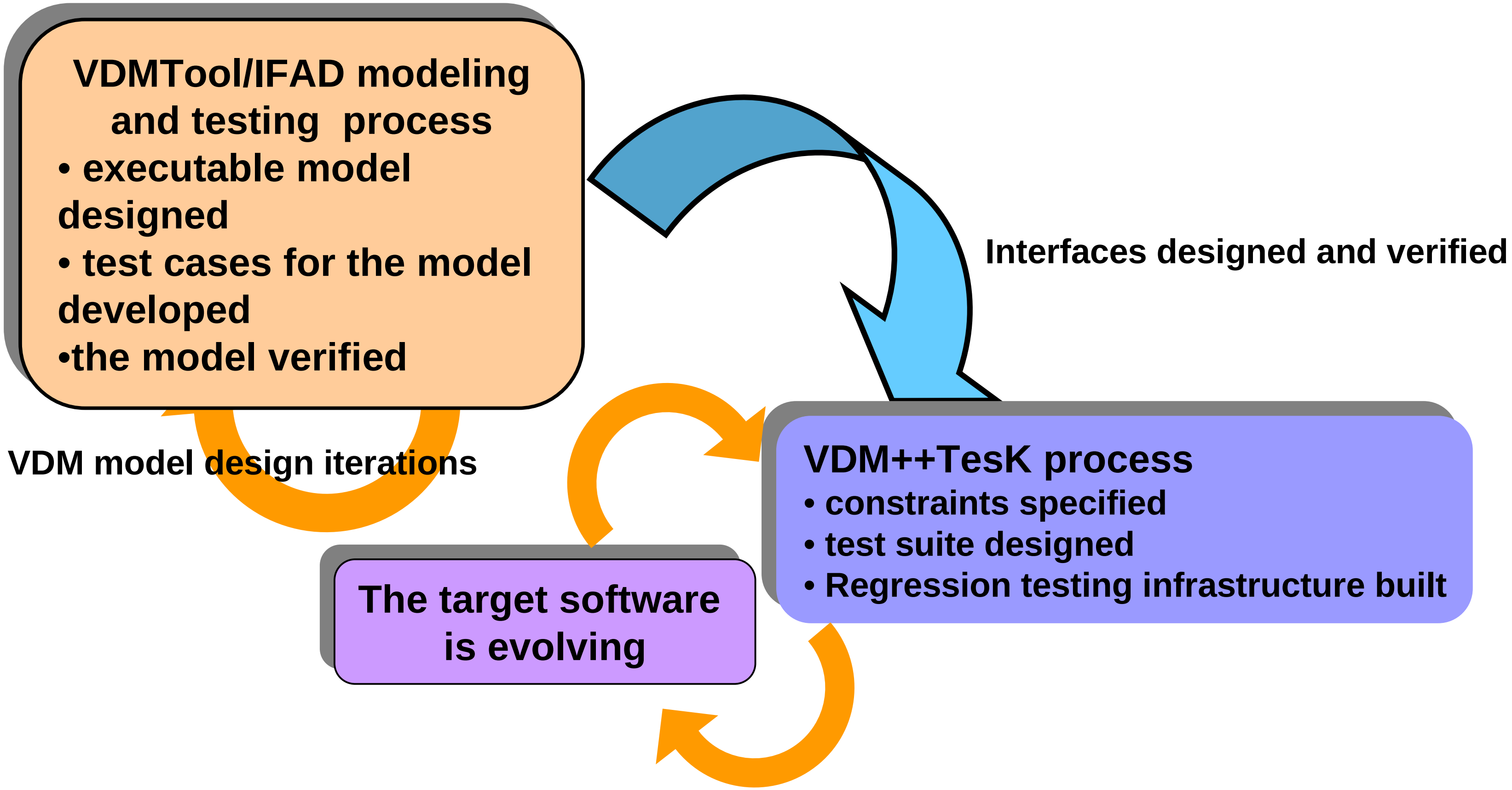
# PROSPER Case studies

- Alarm
- Tracker
- Safer
- Line database (RTRI)
- Interlocking (RTRI)

# Development Guidelines for RT



# IFAD and RedVerst processes integration



VDMTool/IFAD modeling and testing process
- executable model designed
- test cases for the model developed
- the model verified

Interfaces designed and verified

VDM model design iterations

VDM++TesK process
- constraints specified
- test suite designed
- Regression testing infrastructure built

The target software is evolving

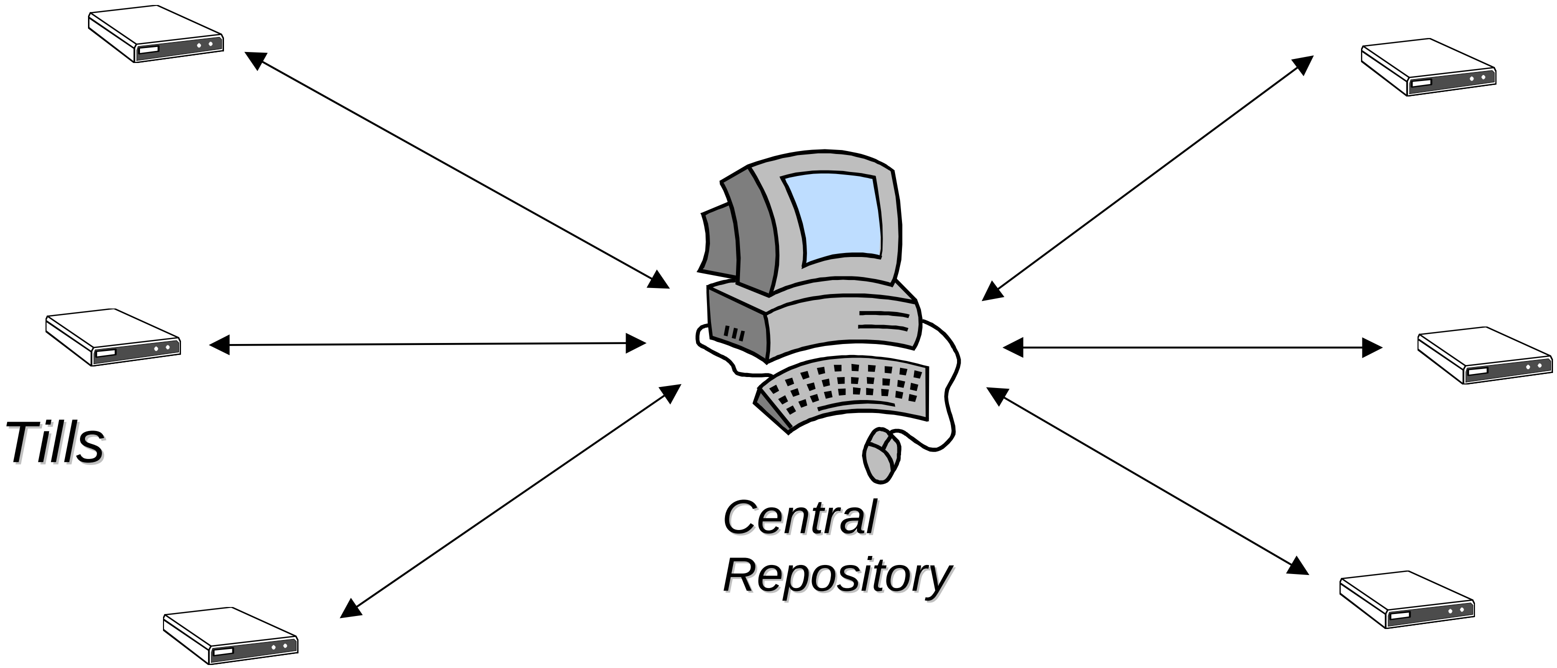
# VDMTools® Tutorial

- ✓ IFAD Profile
- ✓ Where does VDM fit in?
- ✓ VDM++ Overview
- ✓ Overview of **VDMTools**®
- ➢ **Demonstration overview**

# The Cash Dispenser Model

- Model of a system of tills and a central resource.
- Customers interact with tills by inserting a card and entering a PIN
- Central resources contains detailed records of customers’ bank accounts
- “Illegal” cards are kept by the till.

# A Cash Dispenser Example



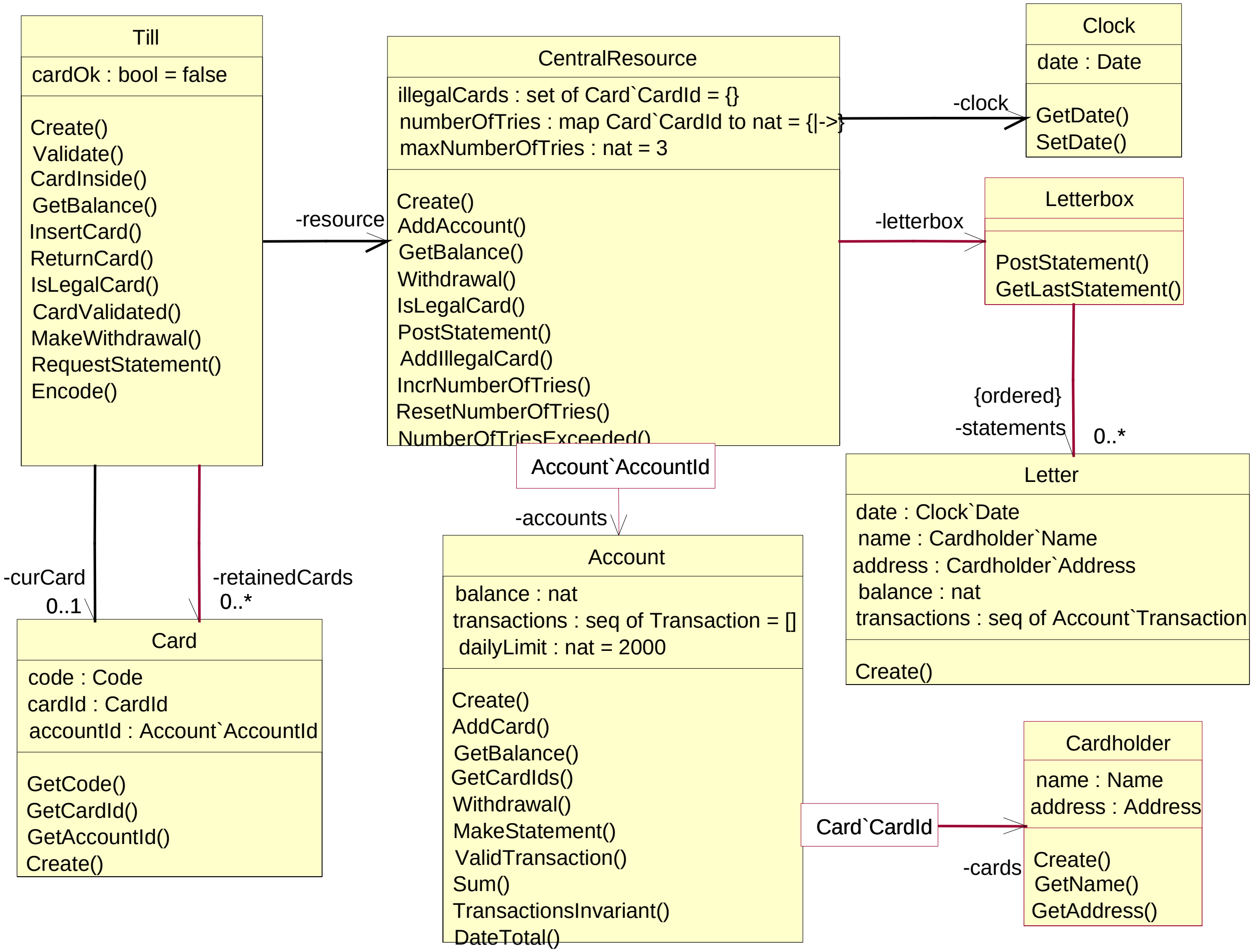
# Requirement Specification

There are many tills which can access a central resource containing the detailed records of customers' bank accounts. A till is used by inserting a card and typing in a PIN (Personal Identification Number) which is encoded by the till and compared with a code stored on the card.
After successfully identifying themselves to the system, customers may try to:
1. view the balance of their accounts
2. make a withdrawal of cash
3. ask for a statement of their account to be sent by post.
Information on accounts is held in a central database and may be unavailable. In that case 1) above may not be possible. If the database is available, any amount up to the total in the account may be withdrawn, subject to a fixed daily limit on withdrawals. This means that the amount withdrawn within the day must be stored on the card.
"Illegal" cards are kept by the till.

# Development Process

- Analysis (using VDM-SL with API animation)
  - alternative to use cases
  - abstraction from multiple tills
- Design (using Rose VDM++ Link with systematic testing and API animation)
  - abstraction from possible failures of tills
- Implementation (with concurrent VDM++ model and automatic Java code generation combined with user interface)

# UML Class Diagram



Till

cardOk : bool = false

Create()
Validate()
CardInside()
GetBalance()
InsertCard()
ReturnCard()
IsLegalCard()
CardValidated()
MakeWithdrawal()
RequestStatement()
Encode()

-resource

CentralResource

illegalCards : set of Card`CardId = {}
numberOfTries : map Card`CardId to nat = {|->}
maxNumberOfTries : nat = 3

Create()
AddAccount()
GetBalance()
Withdrawal()
IsLegalCard()
PostStatement()
AddIllegalCard()
IncrNumberOfTries()
ResetNumberOfTries()
NumberOfTriesExceeded()

-clock

Clock

date : Date

GetDate()
SetDate()

-letterbox

Letterbox

PostStatement()
GetLastStatement()

{ordered}
-statements
0..*

Letter

date : Clock`Date
name : Cardholder`Name
address : Cardholder`Address
balance : nat
transactions : seq of Account`Transaction

Create()

-curCard
0..1

-retainedCards
0..*

Card

code : Code
cardId : CardId
accountId : Account`AccountId

GetCode()
GetCardId()
GetAccountId()
Create()

Account`AccountId

-accounts

Account

balance : nat
transactions : seq of Transaction = []
dailyLimit : nat = 2000

Create()
AddCard()
GetBalance()
GetCardIds()
Withdrawal()
MakeStatement()
ValidTransaction()
Sum()
TransactionsInvariant()
DateTotal()

Card`CardId

-cards

Cardholder

name : Name
address : Address

Create()
GetName()
GetAddress()

# Further Information

- VDMTools brochures
- Download all VDMTools documentation and executables from http://www.ifad.dk/Products/VDMTools/executables.htm
- Toolbox Newsletters available at http://www.ifad.dk/Newsletter/index.htm
- Features described at: http://www.ifad.dk/Products/VDMTools/features.htm