SPECIFIC TARGETED RESEARCH PROJECT
INFORMATION SOCIETY TECHNOLOGIES

FP6-IST-2005-033606

VIsualize all moDel drivEn programming

VIDE

WP 11	Deliverable number D11.3 **Workshop, competition, training course and verification**

Project name:	Visualize all model driven programming
Start date of the project:	01 July 2006
Duration of the project:	30 months
Project coordinator:	Polish-Japanese Institute of Information Technology
Leading partner:	Polish-Japanese Institute of Information Technology
Due date of deliverable:	30.11.2008
Actual submission date	14.01.2009
Status	developed / draft / **final**
Document type:	Report
Document acronym:	DEL

Project supported by the European Commission within Sixth Framework Programme

Editor(s)	Sheridan Jeary, Piotr Habela
Reviewer(s)	Anis Charfi
Accepting	Kazimierz Subieta
Location	www.vide-ist.eu
Version	1.0
Dissemination level	**PU**/PP/RE/CO

Abstract:

The VIDE project aims at a visual, Unified Modeling Language (UML) compliant action language, the VIDE language, suited to business applications. The language is to be used in the model driven software development process (which raises the requirements of its standard-compliance). Further development of the project also includes the integration of a business oriented modelling, aspect-oriented facilities, and means for quality assurance provided inside a powerful, platform-independent development toolset.

This deliverable is intended to describe the evaluation results of VIDE concepts and the software prototype against measurable results and other desired features. In addition, it includes a comparison with other tools available on the market. The evaluation consists of the "vertical" view – where the tool chains available are compared with the VIDE toolset as a whole, and the "horizontal" view – where particular features are evaluated in terms of the benefits they bring individually to particular modelling steps. Most of the horizontal evaluation work was performed in the form of workshops using the VIDE prototype software with participants from outside the project team.

The VIDE consortium:

Polish-Japanese Institute of Information Technology (PJIIT)	**Coordinator**	**Poland**
Rodan Systems S.A.	Partner	Poland
Institute for Information Systems at the German Research Center for Artificial Intelligence	Partner	Germany
Fraunhofer	Partner	Germany
Bournemouth University	Partner	United Kingdom
SOFTEAM	Partner	France
TNM Software GmbH	Partner	Germany
SAP AG	Partner	Germany
ALTEC	Partner	Greece

History of changes

Date	*Version*	*Author*	*Change description*
01.10.2008	0.1	SAP	Document created
03.11.2008	0.2	PJIIT	Agreed structure changes applied
22.12.2008	0.3	BU, SAP	Vertical evaluation input integrated into the document
10.01.2009	0.4	BU, FIRST, IESE, PJIIT, SAP	Horizontal evaluation input integrated into the document
12.01.2009	0.5	IESE	Vertical evaluation updated
13.01.2009	1.0	PJIIT	Final editing

Table of Contents

List of figures

List of tables

1 Introduction

Work Package 11 completes the implementation work of the VIDE project with a practical evaluation of the software solution that has been developed. The evaluation work uses the results of all preceding packages – especially the software prototypes developed in the course of WP9, and investigates their features against the criteria set out in the project's Description of Work document and further elaborated in Work Package 1.

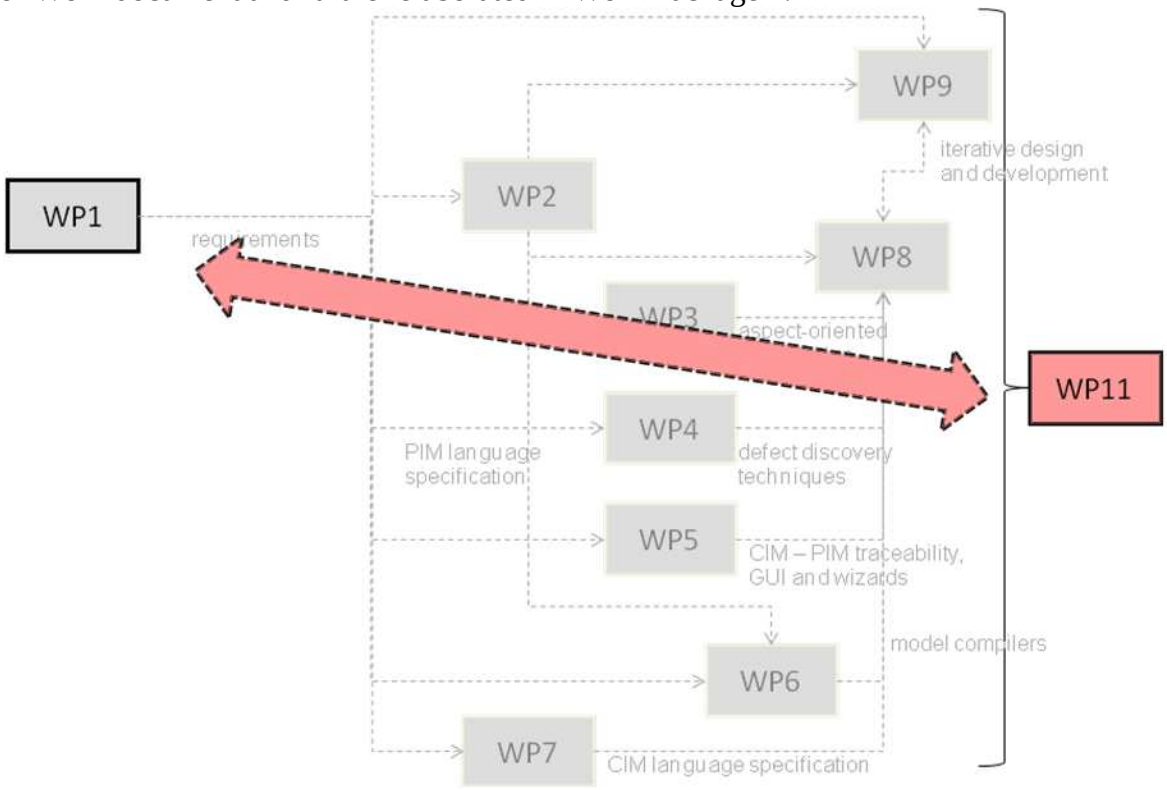

Figure 1: Role of the work package 11 in the structure of VIDE project

Because the goal of the VIDE project was to raise the productivity of software development through the introduction of an integrated, platform independent toolset, the evaluation is focused on:

- investigating the potential productivity gains from individual VIDE component solutions,
- identifying the advantages of the VIDE working as a uniform modelling suite covering MDA's CIM and PIM layers and beyond (vertical integration).

It is necessary to note however, that not all advantages of the tooling could be evaluated this way, as various limitations result from the prototype nature of the evaluated software, in addition to the limited timeframe (hence for example the long-term gains for model maintenance are not considered here).

The evaluation work described in this deliverable does not produce any new component of the VIDE, but instead investigates the toolset as a whole (see Figure 2), with special focus on VIDE modelling tool components (marked with bold line) that were also the subject of individual horizontal evaluation.

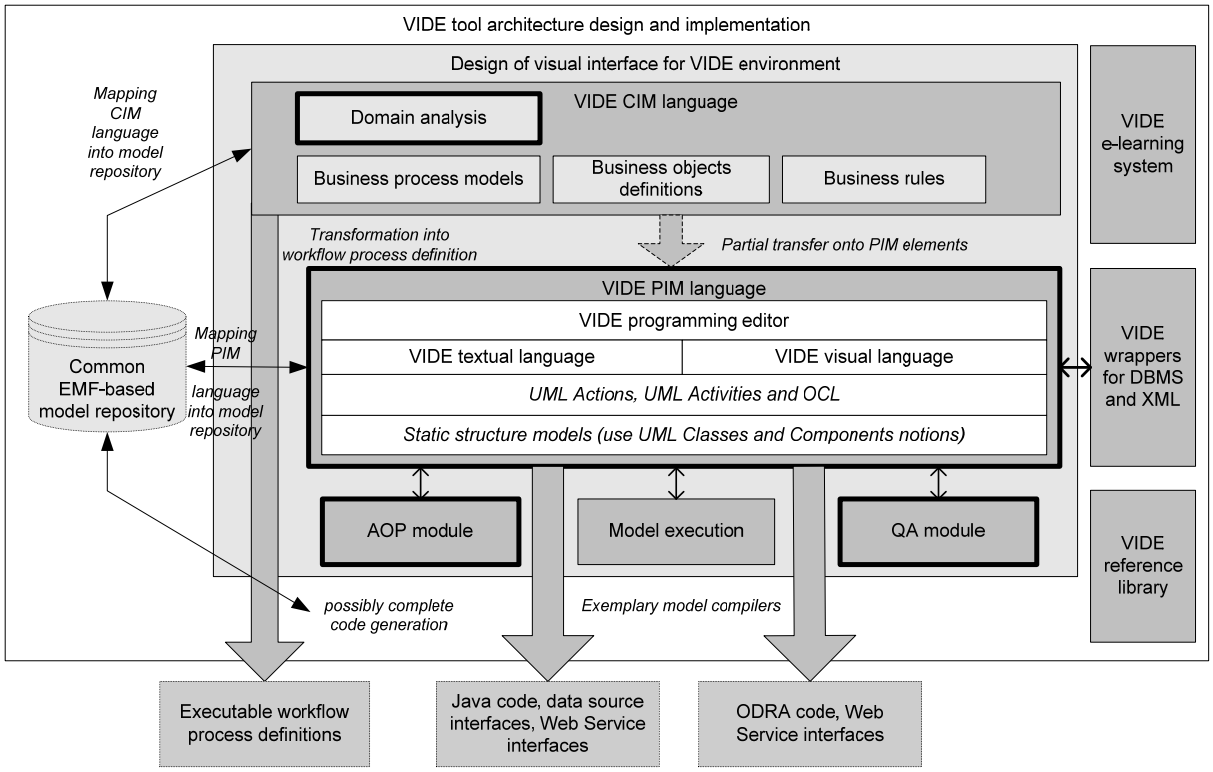

Figure 2: Main subjects of WP11 evaluation work

2 VIDE evaluation approach

The direct aim of the evaluation is in verifying the Measurable Results formulated at the outset of the project (VIDE-Annex, 2005) and other essential features created in the course of research and implementation. These features were to be benchmarked against the tools available on the market. However, it has been necessary to take into consideration the prototype nature of VIDE software where the usability and overall productivity were compared.
Table 1 specifies the Measurable Results as seen at the beginning of the project.

Table 1: Measurable results as specified in the Description of Work document

Measurable result number	Domain	Measurable result	How will it be verified?
1	Visual coding on the PIM level	With VIDE, at least 90% of the code of applications covered by Executable UML will be coded visually on the PIM level.	Typical real-life applications covered by Executable UML will be defined and VIDE as well as the other leading tools we be used to implement them. The amount of code delivered fully visually will be measured and compared.
2	Application development time	Applications covered by Executable UML will be developed with VIDE in 1/2 of the time needed now using both Executable UML tools or more traditional Rapid Application Development (RAD) tools.	The development time of applications described above will be measured and compared.
3	AOP composition at PIM-level	Expressiveness in terms of joinpoint models and possibilities of structural and behavioural adaptation compared to existing AOP approaches. The (visual) aspect composition at PIM-level will be sufficient to cover at least 90% of typical AOP-scenarios in those parts of an application, that can be defined in the VIDE language.	The achieved AO modeling and composition facilities will be compared with existing AOP approaches at programming language level, considering in particular the adaptation semantics and expressiveness at PIM level. Representative scenarios will be chosen for the comparison with respect to the business application domain.
4	Quality of Platform Independent Models	With the discovery of quality defects and following model refactorings the qualitative and quantitative quality as measured against a specific quality model (probably based on ISO 9126/25000 but for MDA) will increase by at least 20%.	The quality of PIM developed with and without the quality defects discovery will be analyzed, measured, and compared. Measurement will be conducted qualitative as well as quantitative using existing design metrics and new model metrics.
5	Skill level needed to use the programming tool	The required skill level is the basic knowledge of a UML-tool. Individuals who have the knowledge from a 1 week preliminary UML course will need 1 week of training to be able to create small Executable UML applications.	The individuals with basic UML tool knowledge but no programming knowledge will participate in 1 week courses ▪ of VIDE ▪ of other leading programming languages and environments Then the abilities of the participants to program simple business systems will be compared. A report summarizing the students achievements and their opinions about VIDE will be delivered.

Measurable Results 1, 2 and 5 were evaluated in the course of two workshops devoted to VIDE PIM-level language and editors functionality:

- VIDE Textual Editor and Visual Expression Builder workshop,
- VIDE Visual Editor workshop.

In both cases a programming assignment was given to a group of individuals from outside the project team and the results were compared between VIDE and traditional programming tools.

Measurable Result 3 was evaluated by a feature analysis, while the Measurable Result 4 was evaluated through a dedicated workshop and questionnaire.

Apart from the planned features reflected in the Measurable Results, VIDE evolved and was extended towards a broader CIM-level functionality. To consider it in the evaluation, an additional workshop was conducted to investigate the potential benefits of the business user oriented features offered by the Domain Analysis Tool.

Another area identified during the project, was the fact that VIDE covers several modelling layers and attempts to achieve uniformity, traceability and automation between them, thus bringing another potential advantage: the combination of a number of features into a single toolset to make them usable in a common development process. Hence the evaluation work has been complemented by the "vertical" view, where the relevant tool chains for complete development processes available on the market were compared with VIDE according to a number of features identified.

The approach is illustrated in Figure 3. The "horizontal" evaluation is mainly (but not exclusively) concerned with the PIM level. The "vertical" evaluation provides an analysis of 27 essential features of VIDE against 10 tool chains available on the market.

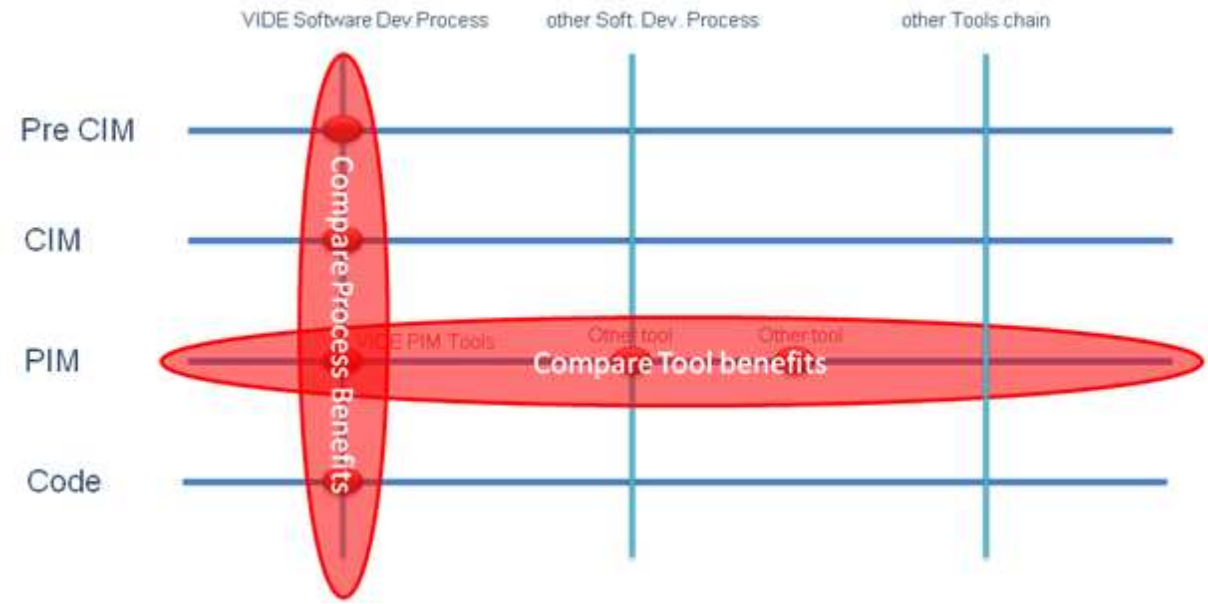

Figure 3: Two dimensions of VIDE evaluation

3 Horizontal Evaluation

In the course of horizontal evaluation, individual features considered the most specific and vital for VIDE have been evaluated in the course of workshops using VIDE software or through comparison analysis. The main subject of the evaluation consisted of the five Measurable Results formulated at the outset of VIDE project that are focused on the CIM level plus the pre-CIM solution that emerged during the project as a new functionality.

3.1 VIDE PIM language evaluation workshop (PJIIT)

3.1.1 Workshop Goals

The workshop was intended to present the VIDE language and tool to a group of students and to verify if the measurable results have been achieved. It consisted in presenting our students on two different levels of studies with a data-intensive programming task which will be described in the next section. The main goal of the workshop was gaining additional experience with using the PIM-level VIDE language and to collect feedback from users with different level of expertise. Apart from that, the workshop was intended to verify the measurable results 1, 2 and 5 specified in the Description of Work (VIDE-Annex, 2005).

1) With VIDE, at least 90% of the code of applications covered by Executable UML will be coded visually on the PIM level.

2) Applications covered by Executable UML will be developed with VIDE in 1/2 of the time needed now using both Executable UML tools or more traditional Rapid Application Development (RAD) tools.

5) The required skill level is the basic knowledge of a UML-tool. Individuals who have the knowledge from a 1 week preliminary UML course will need 1 week of training to be able to create small Executable UML applications.

The students participating in the evaluation workshop were divided into a number of groups working with various tools: traditional programming language, VIDE textual and VIDE visual. We recorded the completion rate of the project as well as the time needed to complete each subtask. This way we gained data to assess the measurable result 1 (by comparing the teams working with visual and textual VIDE tools), the measurable result 2 (by comparing the teams working with VIDE and traditional tools) and the measurable result 5 (by selecting students with basic UML knowledge and providing them with a short tutorial on using VIDE). The students were selected from undergraduate (IT and Management faculties) and graduate groups (IT, Software Engineering profile) to find out, how the general knowledge of query languages influences the result of the workshop exercise. Additionally, in the graduate group a realisation of the workshop assignment using pure ODRA platform was tested in order to evaluate ODRA prototype and to check, how much of the VIDE productivity results from a seamless programming / query language use that is also a feature of ODRA.

The number of students involved and the scope of the assignments do not allow to draw precise conclusions in terms of quantifiable values, nevertheless they provide important feedback from outside the VIDE project team.

3.1.2 Workshop design

3.1.2.1 Team composition

The workshop was devised for four kinds of team (2 to 3 students each) which will be called using the names of the applied tools.

1. VIDE visual editor (2 undergraduate teams and 1 graduate team).
2. VIDE textual editor (3 undergraduate teams and 2 graduate teams).
3. Java/SQL (1 undergraduate team and 1 graduate team).
4. ODRA (1 graduate team)

We also conducted the same experiment with two developers working for PJIIT within the VIDE project. Both of them know the tool very well, although they did not have much experience in writing applications in VIDE. This experiment allows comparing the productivity of a skilful VIDE user and a team of newbies.

3.1.2.2 Comparison to other tools for action modelling

One of the comparisons assumed at the beginning of the project for Measurable Result 2 was the development time measured for VIDE and "other Executable UML tools". Initially, respective comparison to Topcased or IBM Rational Software Modeler activity and action visual editor was considered, because these tools are based on UML 2 –compliant metamodel implementations. However, due to substantial limitations of those features in abovementioned tools (see section 4.4.4 for details), it was impossible to conduct direct comparison. The most important limitations include:

- incomplete support of activities, in particular – Structured Activities (for example – lack of support for SequenceNode. This would enforce using flow-based modelling and hence following a radically different style of representing method behaviour
- lack of features available that would allow a formal and fully transformable to metamodel instance representation of ConditionaNode or DecisionNode
- very difficult and error-prone editing of behaviour due to the lack of visual distinction among different kind of action constructs and very limited contextual hints.

However, it is necessary to note that even in case of improvement of those editors in terms of differentiated visual symbols and contextual hints, but maintaining the existing approach (that assumes 1:1 visualisation of respective UML metamodel elements) the development would rather still suffer from too fine-grained style of specification.

To illustrate that problem, consider the example of a simplified method behaviour shown below, that was possible to specify using Topcased (note the method intentionally avoids using conditional statements).

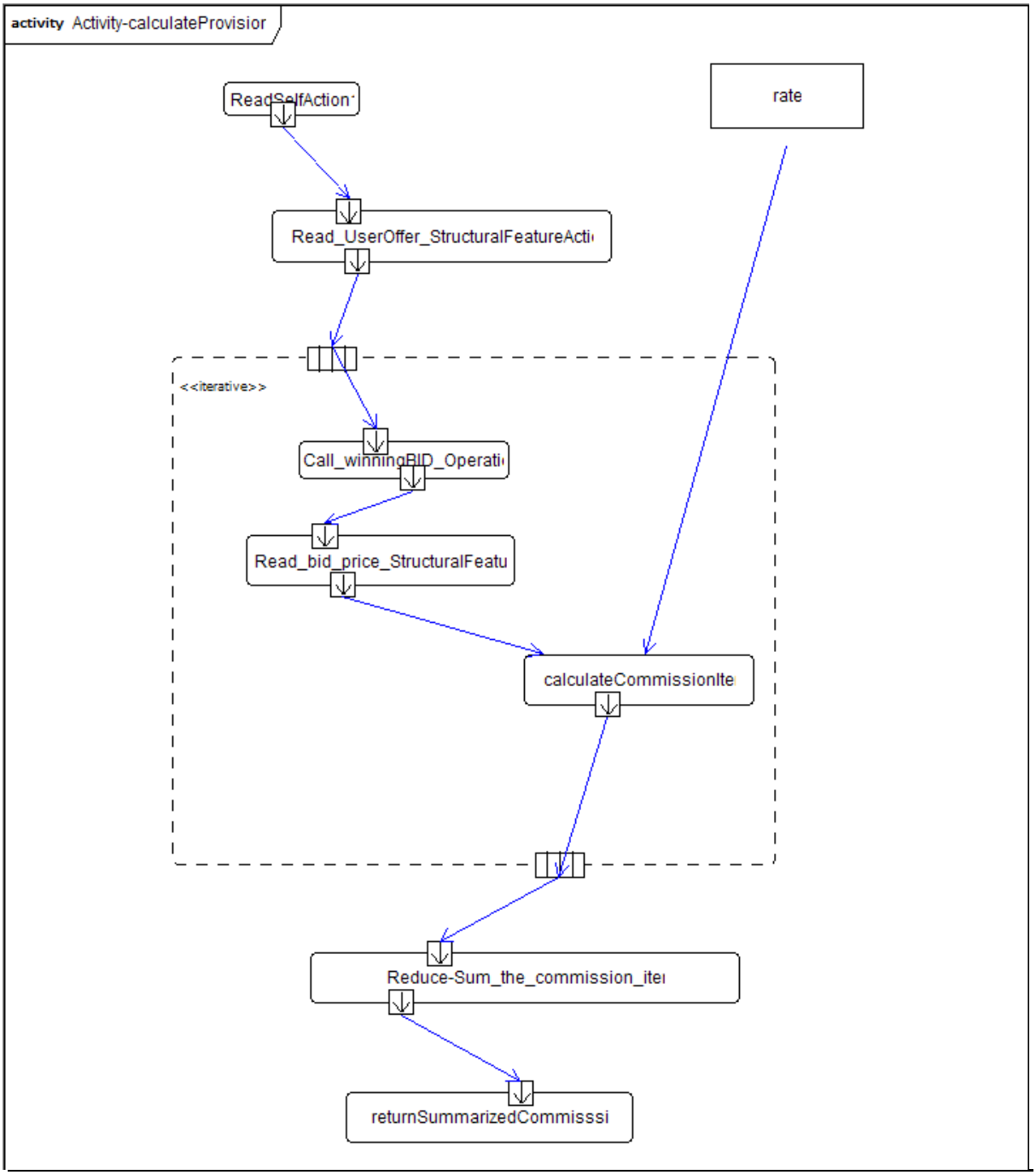

Figure 4: Topcased action editor specified sample of the "calculateProvision()" method's behaviour

To draw this, it was necessary to put o the diagram 21 nodes of appropriate kind, assign their names, specify the types for most of them (input and output *pins* in particular) and connect them with 9 flows. Due to the current limitations of the Topcased editor it was easy to pick mutually incompatible types or to refer to a feature belonging to a different class but having the same name as the intended one. Moreover, even with this amount of diagram constructs, the behaviour is not yet specified formally – as the algorithms of calculating the commission (rate * bidPrice) and the way of reducing collection of commission items (sum) were set here only as comments to appropriate action nodes. Respective code for VIDE textual language – having a complete mapping onto UML 2 and OCL 2 metamodel is:

```
provision : Real = 0.0;
offer.winningBid() foreach { b | provision:= b.price * rate; }
return provision;
```

Similarly, the same behaviour can be specified in VIDE visual editor with following constructs:

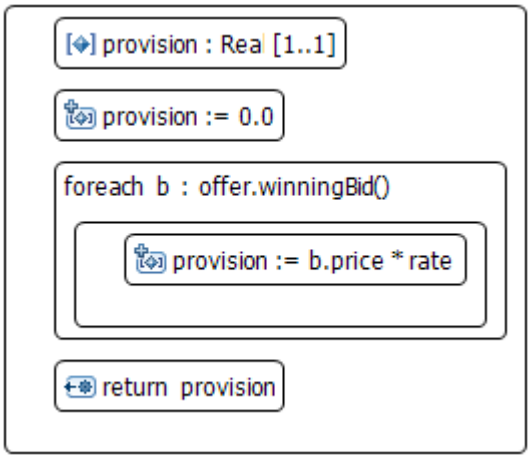

Figure 5: VIDE Visual Editor specified sample of the "calculateProvision()" method's behaviour

To sum up – limitation of alternative action editors did not make it possible to conduct a workshop evaluation for this kind of comparison. However, based on the above small example and the structural characteristics of abovementioned editors (e.g. – the number of elements that need to be created, named and connected), authors if this report expect VIDE could be at least 2-3 times faster than Topcased action editor in its current

3.1.2.3 Introductory Tutorial

First, a short tutorial (120 minutes) has been conducted and a simple example was coded with all the teams. Apart from that tutorial and their previous knowledge of UML (especially – class diagram modelling) – the students were only provided with three short manuals 5-10 pages each (written in Polish), devoted to textual VIDE language, visual VIDE language and Visual Expression Builder. The following "Simple Shop" example was built during the tutorial. Below the data model of the shop together with the description of behaviour (methods) can be found.

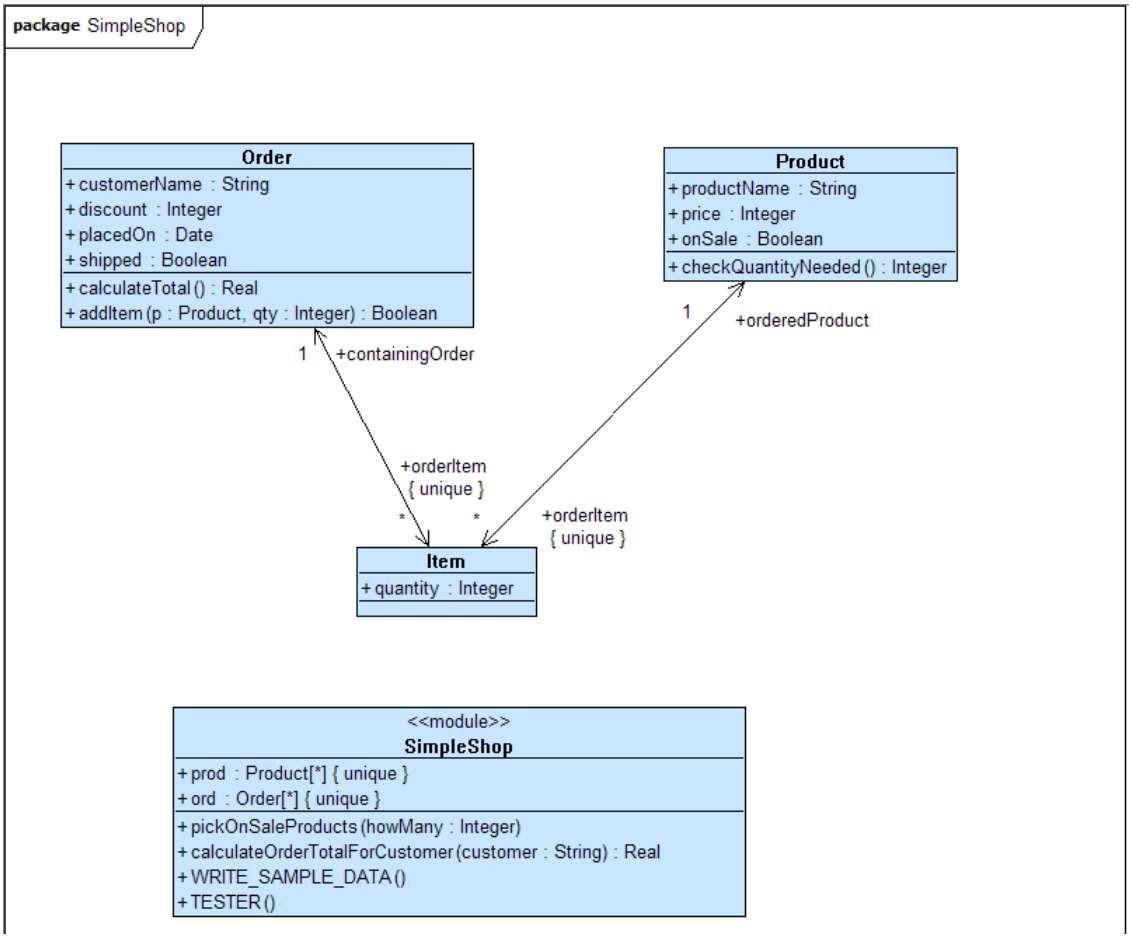

Figure 6: UML class model for the workshop's introductory tutorial

Order.calculateTotal() : Real

Return the total value of the order after subtracting the **discount**.

Order.addItem(p: Product, qty : Integer) : Boolean

Add a line item for **qty** pieces of product **p**.

SimpleShop.pickOnSaleProducts(howMany : Integer)

Return **howMany** cheapest of products currently **onSale**.

SimpleShop.calculateOrderTotalForCustomer(customer : String) : Real

Find the customer and return the total net value of his/her orders (hint: use the method **Order.calculateTotal**()).

Product.checkQuantityNeeded() : Integer

Return the quantity of the product in all orders which are still not shipped.

3.1.2.4 Experiment on application development

The real experiment (120 minutes) was conducted using the “Auction Site” example with the following structure model.

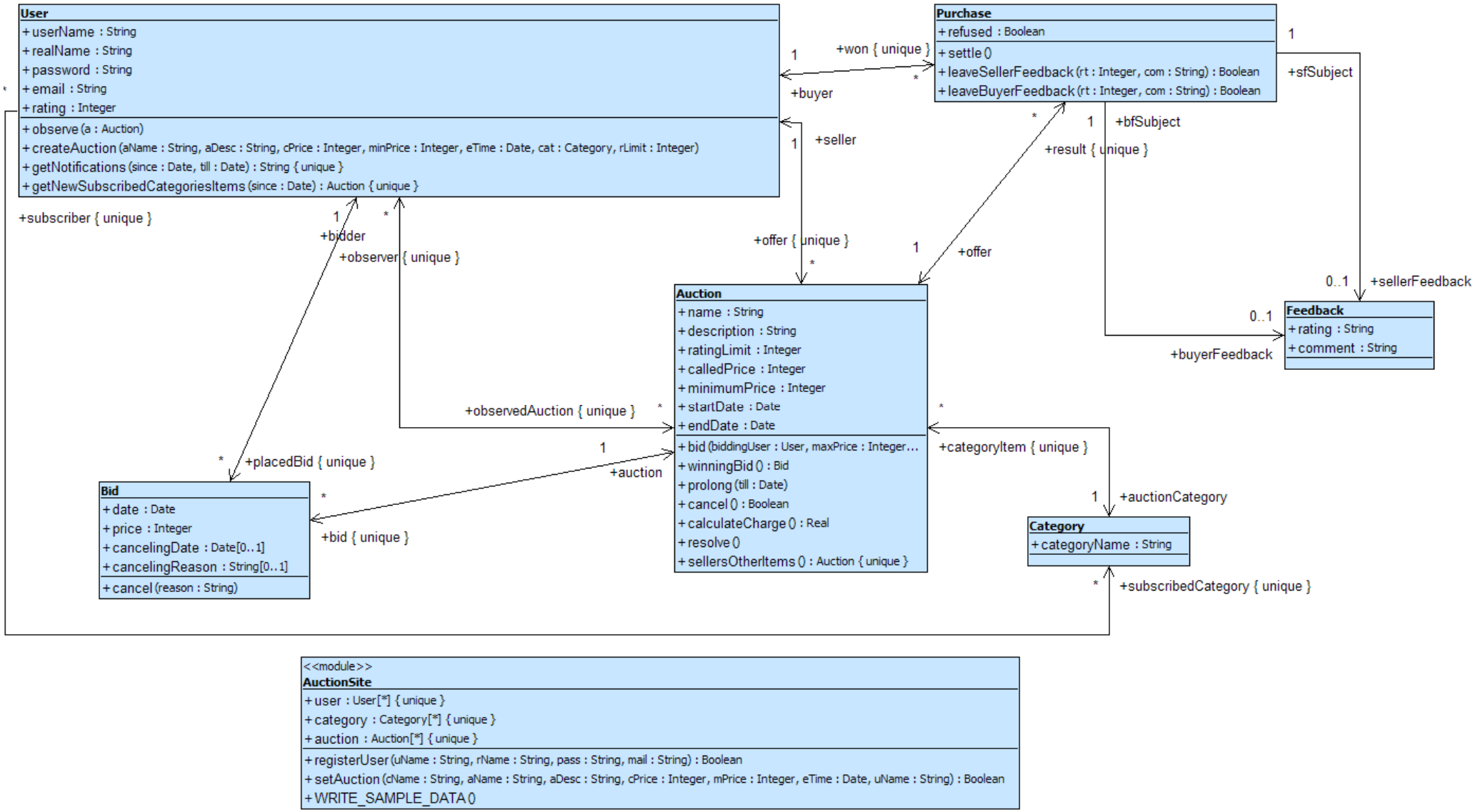

Figure 7: UML class model for the workshop's experimental assignment

Students were to create implementations of the following behaviour (methods).

User.observe(a : Auction)

Creates **observedAuction** link to the **a** auction. Before this – checks if the auction is not linked already to avoid duplicates.

User.createAuction(aName : String, aDesc : String, cPrice : Integer, minPrice : Integer, sTime : Date, eTime : Date, cat : Category)

Creates new `Auction` object, filling its respective mandatory attributes (`aName`, `aDesc`, `cPrice`, `minPrice`, `eTime`, `rLimit`), and linking it to the target user (**`self`**) and category provided as a parameter (`cat`).

User.getNotifications(from : Date, till : Date) : String [0..*]

Returns a collection of strings, each formulated as one of the following (only the events dated between from and till are to be included):

- New bid in your auction <auctionName> placed on <bid_date>. Current price <amount>
- Your auction <auctionName> ended - winning offer by <userName> at price <amount>
- Your auction <auctionName> ended with no sufficient bid placed.
- You won auction <auctionName> with price <amount>

User.getNewSubscribedCategoriesItems(since : Date) : Auction [0..*]

Returns a collection of `Auction`s that belong to the categories connected by a `subscribedCategory` link to the current `User` and that began after the `since` date.

Bid.cancel(reason : String)

Set optional fields for this `Bid` object – `cancelReason` and `cancelingDate` (the latter using the current date read from system).

Auction.bid(bidder : User, maxPrice : Integer) : Boolean

Check if the User's `rating` is greater or equal to `ratingLimit`. If not – return false;

Otherwise, create a `Bid` object, filling its date and price fields with current system date and `maxPrice` parameter values respectively. Connect this object with `bidder` link to the `User` and with auction link to the `Auction`. Invoke the `winningBid()` method and calculate the result to be returned. If the result is equal to the currently created `Bid` object AND the price offered matches the `Auction`'s minimum price – return true. Otherwise return false.

Auction.winningBid() : Bid [0..1]

Select the bids that are not cancelled and sort them according to their dates and then – according to their `price` (descending) and return the first item as the operation's result.

Auction.prolong(till : Date)

Check if the new `till` date is indeed later than the one currently set. If so – update the `endDate` field.

Auction.cancel()

Check if the current date is earlier than Auction's `endDate`. If so, do the following: For each `Bid` placed invoke its `cancel` operation setting "Auction cancelled by seller".

Auction.calculateCharge() : Real

Calculate the `base` value as the maximum of `minimumPrice` and `calledPrice`. If the value is lower than 5 – return 0.2, if lower than 20 – return 0.8, if lower than 100 – return 5, otherwise – return `base` * 0.04.

Auction.resolve()

Invoke the `winningBid` operation to determine the best bid. If a non-empty result was returned, check if its price is higher than the `minimumPrice` of the Auction. If so, create a `Purchase` object and connect it with its links to `buyer` (retrieved from the Bid). Connect the Auction's `result` link to the `Purchase` created.

Auction.sellersOtherItems() : Auction [0..*]

Determine the current date. Then return the result of the query that navigates from auction to the seller and from it – back to the Auction using the `offer` link, selecting only those auctions whose `endDate` is later than the current date.

Purchase.settle()

Set the `refused` field to true. Decrease the buyer's `rating` by 20. Then, follow the algorithm like in `Auction.winningBid() : Bid [0..1]`, but here additionally filter out from the result the bid or bids by the refusing buyer. If another satisfactory bid is determined this way – create the `Purchase` object for respective user – analogously like in `Auction.resolve()`.

Purchase.leaveBuyerFeedback(rt : Integer, com : string) : Boolean

Check if the `buyerFeedback` link already contains some `Feedback`. If so – return false. Otherwise – create respective `Feedback` object, provide values of its fields from parameters and connect it to the `Purchase` with the `buyerFeedback` link. Update the `rating` field of the seller (you need to determine the proper object by navigating through the `Auction`) by the value of the rating (it is expected to be 1 for "positive", 0 for "neutral", or -1 for "negative"). Return true.

Purchase.leaveSellerFeedback(rt : Integer, com : string) : Boolean

Perform the steps for the buyer and `sellerFeedback` link analogously like in case of the above method `leaveBuyerFeedback`. The only structural difference is that determining the buyer object is more straightforward, as it is directly linked to the `Purchase` object.

AuctionSite.registerUser(uName : String, rName : String, pass : String, mail : String) : Boolean

Check in the user property for existence of an object that already contains `userName` equal to the `uName` parameter – if found, return false. Check in the `user` property for existence of an object that already contains email equal to the mail parameter – if found, return false. Otherwise, create a `User` objects filling its fields with respective parameters and store it in the user property. Return true.

AuctionSite.setAuction(cName : String, aName : String, aDesc : String, cPrice : Integer, mPrice : Integer, eTime : Date, uName : String) : Boolean

Locate in the `user` property a user with `userName` equal to `uName` parameter. If not found or if the user `rating` is lower than 0 – return false. Locate in the `category` property a category with `categoryName` equal to `cName`. If not found – return false. Invoke the `createAuction` operation on the User object retrieved, providing respective parameters to it. For the rating limit (`rLimit` parameter) assume 0 by default. Return true.

As can be seen, due to limited duration of the workshop, the structural part of the application was provided as ready-made in order to focus the experiment on the features that are specific for VIDE – namely, the behaviour specification.

The teams implementing their assignments in VIDE got the Topcased UML class model as the starting point. The teams working in Java and SQL were provided (apart from the UML diagram) with SQL schema specification. Similarly the team working with ODRA got the UML diagram and ODRA module definition skeleton generated using VIDE ODRA model compiler.

3.1.2.5 Experiment on query writing

While the previous part of the workshop was aimed at testing the overall functionality of the VIDE language and the technologies being compared with it, the last part was intended to provide more insight into the expression part of respective languages. This was realised by writing ad-hoc queries. The assignments for respective teams were as follows:

- VIDE Textual evaluation teams ► OCL queries
- VIDE Visual evaluation teams ► OQBE diagrams offered by VIDE Visual Expression Builder
- Java/SQL evaluation teams ► SQL queries
- ODRA evaluation team ► SBQL (ODRA platform language) queries

The same model as described in previous section was used and provided with sample data. The following queries were to be formulated by students in SQL, SBQL, VEB and OCL:

1. List all users with `rating` above 100.
2. List all users with `rating` above 100 and having set up at least one auction before 2000-11-28.
3. List all non-cancelled bids in auction named ‘VIDE Cookbook’ sort them according to their dates (`bidDate`) and then according to their `price` (descending).

4. List all auction that ended in a period (2008-11-28, 2008-11-30) inclusively and return for each of them a structure containing of fields: `auctionName` (containing auction's name) and `auctionCharge` (containing the result of `calculateCharge()`).
5. For auction named 'VIDE Cookbook' list all bids placed within a period (2008-11-28, 2008-11-30) inclusively. For each bid list the `bidDate` and `offeredPrice`
6. List all auctions that were active in a period (2008-11-28, 2008-11-30) inclusively and have any bids.: For each such auction - list its name and its winning bidder name.
7. List all failed auctions ended after 2008-11-28, i.e. auctions without a sufficient bid.
8. For a user 'Brown' and period (2008-11-28, 2008-11-30) inclusively, list all auction won by this user (take into account only the sufficient price offers)

Subsequent queries in the assignment belonged to "fraud detection" theme and involved more navigation:

9. Find a user who made a bid in an auction such that the seller and bidder have the same `realName`.
10. Find pairs of users who placed bids in each other auctions such that the bids were not the winning ones.
11. Find pairs of users who placed bids in each other auctions such that at least one of those bids were winning and the user refused the purchase.
12. Find triples of users U1, U2, U3, such that U1 placed a bid in an auction of U2, U2 placed a bid in an auction of U3, an U3 placed a bid in an auction of U1.
13. Limit the above queries to "circles" that occur in the 'Other' category entirely.

3.1.3 Results

In the tables below we summarized the result of the workshop. In these tables **BSc** means a team of undergraduate students, **MSc** means a team of graduate students and **Emp** are the two individual VIDE project employees. Four main column depict teams working with VIDE textual editor, VIDE visual editor, ODRA target platform and in Java. The numbers in this table are minutes used to complete the subtask. A dash means the failure of the team in this particular subtask.

Table 2: VIDE PIM language evaluation workshop part 1 results

Behaviour	VIDE textual editor					VIDE visual editor				ODRA	Java		
	BSc	BSc	MSc	MSc	Emp	BSc	BSc	MSc	Emp	MSc	BSc	BSc	MSc
User.observe	1	3	15	3	3	15	10	5	3	20	15	-	-
User.createAuction	20	13	30	5	6	5	15	10	8	40	10	-	-
User.getNotifications	50	55	30	20	61	40	-	-	30	-	-	-	-
User.getNewSubscribedCategoriesItems	-	-	10	3	1	2	-	-	3	5	-	-	-
User.subscribe	-	-	2	1	2	5	-	-	2	2	-	-	-
Bid.cancel	3	1	2	1	1	2	15	5	3	-	-	-	-
Auction.placeBid	8	10	5	5	7	10	10	20	8	10	-	-	-
Auction.winningBid	5	-	5	5	2	5	20	7	7	3	-	-	-
Auction.prolong	1	1	8	1	1	2	-	3	2	3	-	-	-
Auction.cancel	5	1	20	2	2	2	-	10	4	10	-		-
Auction.calculateCharge	5	6	10	1	4	-	-	10	10	7	-	-	-
Auction.resolve	8	-	4	5	8	-	-	15	11	-	-	-	-
Auction.sellersOtherItems	-	-	-	3	1	-	20	7	2	-	-	-	-
Purchase.refuse	-	-	-	10	8	-	-	3	11	-	-	-	-
Purchase.leaveBuyerFeedback	-	-	-	5	8	-	-	-	15	-	-	-	-
Purchase.leaveSellerFeedback	-	-	-	1	4	-	-	-	5	-	-	-	-
AuctionSite.registerUser	13	-	-	5	4	-	-	-	8	10	-	-	-
AuctionSite.setAuction	-	-	-	10	8	-	30	-	13	-	-	-	-

This table shows that the Measurable Result 2 has been met. Apparently both VIDE editors performed much better than Java. Furthermore, it must be noted that the students working in Java had background and experience in this language, while no one has those for VIDE. Moreover, it was not possible to perform a more thorough training in VIDE in the course of the workshop, hence it was much shorter that the "1 week course" suggested in the Measurable Result 2. It make these results more appealing. The ODRA target platform behaved similarly, albeit a little bit worse. The similarity is caused by the fact the ODRA's SBQL is also a high-level programming and query language and shares many features with textual VIDE language. The Measurable Result 1 seems to be met as well, since all the functionality of the workshop assignment model was possible to encode using VIDE Visual Editor solely. However, due to some functionality limitations of the Visual Editor prototype, the teams using it were a little handicapped. The other reason might be that for programmers the textual syntax was more familiar than the visual. Both notations still beat Java and other traditional approaches – not necessarily by the fact of using a visual notational concept but by high abstraction of the language (as this is true both for visual and textual variant of VIDE PIM). The Measurable Result 5 seems to be met as well. There is no dramatic difference between BSc teams and more skilled MSc teams. Yes, the latter performed better, but it does not look like a difference of an order of magnitude. The results from the **Emp** columns have similar interpretation (although experienced VIDE programmers performed better, the difference is not spectacular).

Below you can find a table summarizing the results of the query part. Students worked with VEB (Visual Expression Builder), textual OCL, SBQL and plain SQL. The numbers in this table are minutes used to complete the subtask. A dash means the failure of the team in a particular subtask to complete respective query in the time frame of the experiment.

Table 3: VIDE PIM language evaluation workshop part 2 results

Query	VEB					OCL		SBQL	SQL	
	BSc	BSc	MSc	MSc	MSc	BSc	MSc	MSc	BSc	BSc
1	2	2	5	2	10	1	3	1	1	1
2	10	3	5	10	5	10	10	2	2	3
3	20	10	5	5	10	20	12	10	10	4
4	5	15	10	20	10	2	15	4	4	5
5	10	10	9	1	4	5	40	8	2	2
6	15	10	12	8	10	10	10	15	8	-
7	4	10	5	5	10	6	5	3	2	-
8	3	15	6	5	8	-	11	5	5	-
9	7	-	5	3	5	-	15	7	4	-
10	10	-	15	20	15	-	-	-	-	-
11	20	10	10	20	10	-	-	-	-	-
12	1	-	4	5	5	-	-	-	-	-
13	5	-	4	10	1	-	-	-	-	-

The results are unmistakable. VEB was the most convenient tool for writing queries. In case of queries and VEB the Measurable Result 2 was met. The visual notation for queries outperformed the other. The same remark concerns Measurable Result 2 in case of VEB: all teams working with VEB achieved much better results than all other teams. All textual languages behaved similarly, although only SQL and SBQL (the latter on a basic level only) has been known to students prior the workshop. One could expect that SQL will be the champion here, which definitely is not true. All the textual notations turned out to be difficult in complex fraud-detection queries 10-13. The Measurable Result 5 also seems to be met, since also this part of the assignment was possible to be completed by students after their short training in VIDE.

The Measurable Result 2 was not met in case of sole OCL (which is a part of VIDE textual language). The OCL notation amounted to be comparable with other textual syntaxes. Hence it may be assumed that the boost of programmers' productiveness observed in VIDE comes from the seamless integration of queries and persistence into the language rather than from specific features of its expressions.

3.1.4 Detailed remarks and participants' comments

In this section we present detailed participants' remarks and their interpretation. Most of those remarks are complaints, which is understandable as the students dealt with a software that is a research prototype. Users were presented prototype tools with a number of flaws, while they still achieved better results than using traditional tools. This further emphasizes the results.

3.1.4.1 Textual VIDE editor

Remarks:

- The class definition in the model browser must always be clicked prior to execute a query.
- Facilities of syntax completion and syntax hints are lacking.
- The syntax is sometimes unusual: OCL arrows, Smalltalk-like postfix operators (especially **foreach**).
- Semicolons are inconsistently used. There are places where they are unexpectedly forbidden and places where they are unexpectedly mandatory.
- A database object browser would be helpful.
- The values of nested objects are not displayed in results of queries.

Praises:

- Although the language looks strange, after a short time it turns out to be convenient and speeding the coding process.
- Using VIDE textual language, more emphasis is on the very solutions than on the language itself.
- The workshop showed that one can code applications fast and joyfully focusing on the applications logic and not the language constructs.

3.1.4.2 Visual VIDE editor

Remarks:

- Some inconsistencies occur when switching between textual and visual syntax.
- The way to add items to the diagram is not intuitive.
- The content of textual fields is cleared when a faulty text is entered (one cannot correct it after failed mode verification, because it disappears).
- It would be nice to have the tab key moving the focus around.
- One cannot open more than one visual editor (e.g. to copy and paste items between the windows).
- The copy&paste facility is not working.
- The item names could be more intuitive (it is probably a feature of UML and not the Visual Editor).

3.1.4.3 Visual Expression Builder

Remarks:

- A precise description of VEB semantics is lacking.
- A copy & paste facility for VEB diagrams would be of much help.
- A comparator edge should be also applicable to results of OCL expressions.
- It happens that developed models do not save.
- Deleting the «output» flag is not easy.
- There is no syntactical and semantic check in embedded textual OCL expressions.

For some kinds of queries the advantage of VEB was especially visible. To demonstrate it we attach a solution of one of the assignments on ad-hoc queries. Please find below the VEB query “Find pairs of users who placed bids in each other auctions such that at least one of those bids were winning and the user refused the purchase.” The picture is really intuitive and is almost a direct mapping of the query formulation. Below the picture you can find the generated OCL query, which is rather awkward.

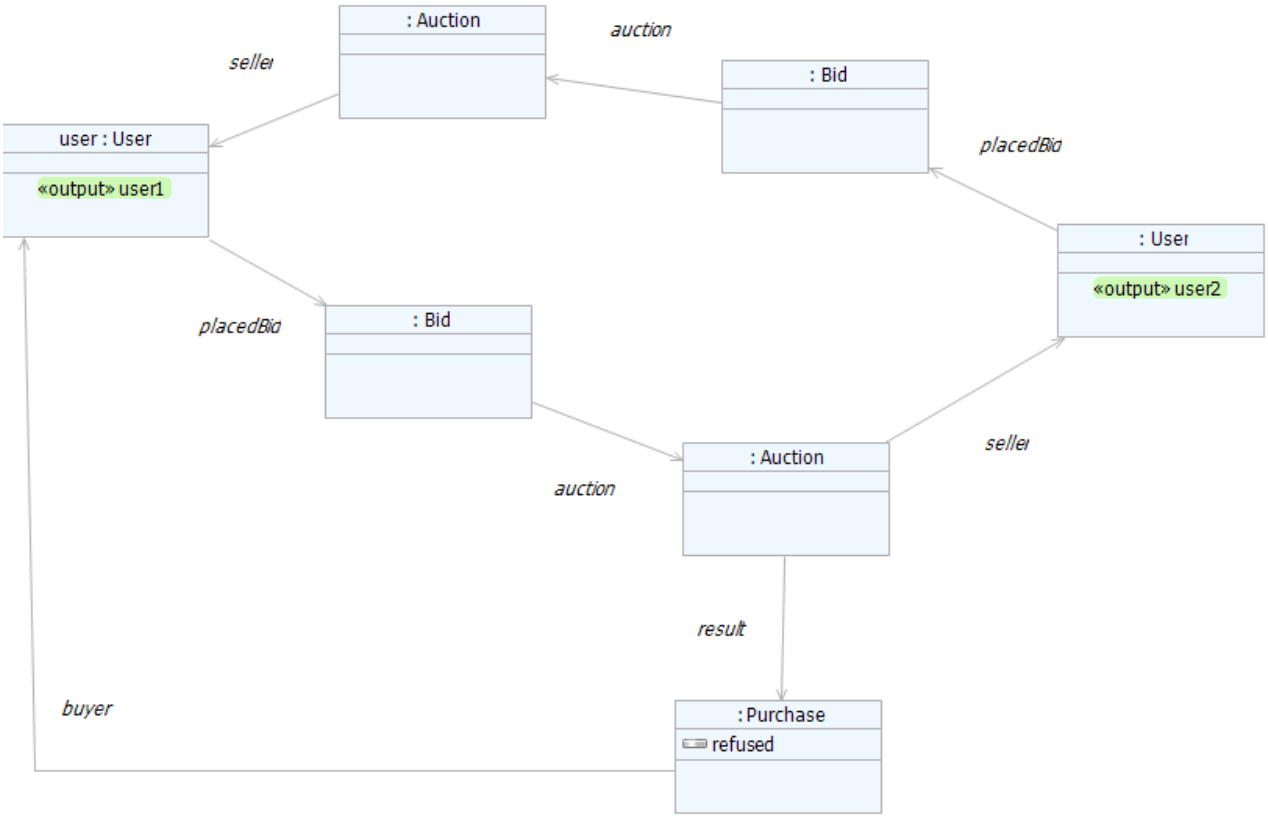

Figure 8: A "Fraud detection" query sample specified using VEB

```
user->collect(u |
            user->collect(v | Tuple{
                  u1 = u,
                  u2 = v}
            )
)
->select(u1<>u2)
->select(u1.placedBid
      ->exists(b |
      (b.auction.seller=u2)
      and (b.auction.winningBid()=b)
      and (b.auction.result->exists(refused))
      )
)
->select(u2.placedBid
      ->exists(y |
            (y.auction.seller=u1)
            and (y.auction.winningBid()<>y)
      )
)
->collect(Tuple{
            user1 = u1.userName,
            user2 = u2.userName
      }
);
```

Figure 9: A "Fraud detection" query sample written in OCL

3.1.4.4 Java / SQL

Remarks:

- The amount of allotted time was too small. The task was not a hard one, but we needed more time to complete it.
- More complex tasks in SQL (ad-hoc queries, 10-13) requires much more effort and time. A visual tool generating SQL queries would be very valuable.
- The framework of the guts of the solution had to be designed from scratch. This caused a lot of problems and cost a lot of work. We would be happy to focus on very business logic.
- The java.sql package is very complex and user-unfriendly.
- There were some problems with the chose NetBeans IDE (it is still being developed and not 100% stable).
- There were problems with Oracle dialect of SQL.
- Lack of skills needed to quick creation a solution framework and rapid switch to implementation.

Praises:

- Java/SQL is a well documented technology.
- Solution to problems can be found easily in literature and Internet.
- There is a lot of free integrated development environments (IDE).

In general the part of the task which consists in method implementation was not even stared. Much time has been used to organize the work, design the persistency solution and implement it. Eventually, a solution framework has been established. The decision was to generate the skeleton using UML designer of NetBeans IDE. All the classes were generated without laborious pounding the code. Furthermore, many time was consumed in order to develop simple façade class responsible for the interaction with the database. Although its simplicity, a poor quality class has been developed, since its user was obliged to cater for ResultSet's closure and SQLException handling. Also the lack of inherent multithreading makes such a solution useless. Adding to the developed code the possibility of running more than one thread safely would require a lot of work.

The participants confirm that Java is not a very good choice as a tool for fast development of data-intense business application. They wondered how would other powerful attested tools behave (like Hibernate) in this setting.

3.1.4.5 ODRA/SBQL

Remarks:

- Lack of mechanisms facilitating coding "**IntelliSense**".
- Problem with unintuitive SQBL ref/deref operators.
- The VIDE method skeleton generator did not produce return statements and braces.
- There were problems with ODRA server – it used to raise OutOfMemory exception.

- Two buttons to execute queries are misleading.
- It's hard to write queries in textual cli.bat client.
- Results of ad-hoc queries were not readable.
- ODRA's error reports are horribly bulky and almost unreadable.
- The cli.bat client allows entering and executing only one query at a time.

3.1.5 Conclusions

Due to the limited amount of time and limited number of participants, the results cannot be interpreted as statistically proven quantifiable values. Nevertheless, interesting observations and feedback from outside the VIDE team were collected. They show that although VIDE is only a limited research prototype, it has a potential to significantly improve the development productivity (at least for less experience users as those who participated in the experiment). The comparison with other technologies and the experiment involving ad-hoc queries writing allow to conclude that the main source of VIDE advantage are:

- seamless integration of query and programming constructs in a single language
- simple design of the language
- integration with visual structure design using UML class diagrams
- assuring data persistency inside the language.

The particular choices of concrete syntax had a much smaller impact, though the contextual hints and validation where available are very helpful. On the other hand, the concept of Object Query By Example offered by the VEB component of VIDE provided significant benefits especially in case of more complex queries involving joins, selections and navigation – at lease for the groups of users who are not query language experts.

Apart from that a number of interesting comments about the VIDE tool design and its desirable features were collected.

3.2 VIDE PIM visual language evaluation workshop (SAP)

In the following, we report on the user study that was conducted at SAP to evaluate the PIM level visual notation of VIDE and the VIDE PIM visual editor. In this study, we had in total 16 participants and it took 60 to 90 minutes for each. This evaluation activity was not planned in the Description of Work.

3.2.1 Workshop Goals

The user study has two goals:

- The main purpose was to find out whether learning the visual notation and the respective editor is easier than learning a new textual programming language and the respective IDE[1]. **A gain in terms of learning time would contribute to Measurable Result 5, as individuals will be able to program simple business applications after a short training.**
- In addition, we wanted to get through the survey first impressions on the gain in application development time when using the visual editor compared to a traditional IDE for a textual programming language. **A gain in terms of application development time would contribute to Measurable Result 2, as applications will be developed faster with VIDE than with a traditional programming language and its IDE.**
- We also had other questions in mind that wanted to verify, e.g., about the kinds of mistakes that are prevented by the visual notation, about the situations where using the visual notation is better, about the user-friendliness of the visual editor, etc.

3.2.2 Workshop Design

The survey had the following underlying hypothesis: "*The participants will learn the visual notation faster and thus need less time to solve the tasks compared to the textual notation*." The main targets of the study are users that do not program frequently (e.g., users with UML knowledge and basic knowledge about the concepts of object-oriented programming or users who learned programming in the past and then concentrated on other tasks, etc.) can learn the visual notation faster than a new textual programming language.

First, we had first to select a textual object-oriented language. Since several users are familiar with C, C++, and Java, we decided to take a less used programming language and specifically **Smalltalk**. We selected **Squeak** as an IDE for development with Smalltalk. Both VIDE and Smalltalk support the Object-Oriented Programming paradigm and they have similar language concepts and features.

After choosing the contender of VIDE, we had to decide on the tasks the participants should solve. As we were especially interested in finding out how fast can the participants learn the language and the notation, we decided to give them simple tasks from the business applications domain. We also observed them and video-recorded their actions while they made their first steps with the new language, trying to solve the tasks. As an indication of how fast each language was learned the time needed for solving the tasks was measured. The correctness of the provided solution was also checked.

3.2.2.1 Participants

In total, we had 16 participants in the study. All of them had some programming experience but that experience varies from basic knowledge to expertise in a specific language. The participants were all familiar with UML as a modelling language but only for static structures;

[1] **I**ntegrated **D**evelopement **E**nvironment

they did not know about UML actions. Moreover, all of them did not know Smalltalk before. About 2/3 of the participants have never used a visual programming language before.

A *within-groups* design was chosen, i.e. every participant solved the same tasks using both languages VIDE and Smalltalk. In this way, fewer participants were required (for the statistical relevance) and the variation of personal programming skills is less of a problem.

To account for the transfer of the solution into the other language, every second participant started with Smalltalk. Additionally, the tasks were designed in such a way that they were not challenging in an algorithmic sense.

3.2.2.2 Schedule

The evaluation took 60 to 90 minutes for each of participant. The schedule was as follows:

- Briefing: The purpose of the study was explained to the participants and the next steps in the schedule were presented.
- Understanding the tasks: The participant read the tasks printed on a sheet of paper and the supervisor ensured that he/she understood the tasks correctly.
- Reading the tutorial for the first language: For every second participant we switched the order of the first language to be learned and used. The tutorial is 2 to 3 pages long and concentrates only on the basic concepts and those concepts that are needed for solving the task
- Solving the tasks using the first language and the respective tool: The participant then starts solving the tasks based on the tutorial.
- Reading the tutorial for the second language
- Solving the tasks using the second language

3.2.2.3 Tasks

In addition to the short tutorial for the visual notation and for Smalltalk, the participants got a 1 page document with three tasks in total. That document is shown in shown in the listing below.

Moreover, the participants were given a framework with the classes needed for the solving the tasks in both Smalltalk and in UML. Thus, the participants can focus on modelling method behaviours. Figure 10: shows a class diagram of that framework that was provided to the participants in VIDE and Figure 11: shows the same framework in Smalltalk.

Your Tasks:

You will learn two programming languages: Smalltalk and the VIDE visual notation for action modelling. . In the following tasks, you will implement or modify methods. Stubs of the methods are provided for you, so you only need to implement the method bodies.

Before you do the following tasks, make yourself familiar with the simple shop scenario given as an UML class diagram.

In the case of the UML activity diagram, all attributes of the classes (presented in the class diagram) are public, so you can access them directly. In the case of Smalltalk, there are methods with the same name as the attributes in the UML diagram to access the instance variables.

Try to finish the tasks as quick as you can and in the given order.

If you are done with a task, ask the supervisor to introduce you to the next task. You can ask the supervisor for advice at any time.

Task 1

Implement the method *calculateOrderTotalForCustomer* in the class *SimpleShop*.

This method returns the total price of all orders made by a certain customer. To compute the total for a customer, the method should iterate over all orders of the *SimpleShop* and sum up the totals of those orders belonging to the customer. Use the method *calculateTotal* of the class Order to compute the total of a single order.

Hints: The calculated total is of the type Integer (only relevant for the VIDE case). The name of the customer is passed as an argument to the method and is named customer. Two customer names can be compared using the operator "=" (in both languages).

Task 2

Modify the method *calculateTotal* of the class *Order* to include only those products in the total of the orders that are on sale.

Hints: whether a product is on sale is determined by the attribute *onSale* of the class *Product*.

Task 3

The method *shipOrders* of the class *SimpleShop* ships all those orders of the shop whose items are all on sale. However, the case that not all items of an order are on sale is handled elsewhere now. Your task is to remove the check whether all items of an order are on sale from the method *shipOrders*.

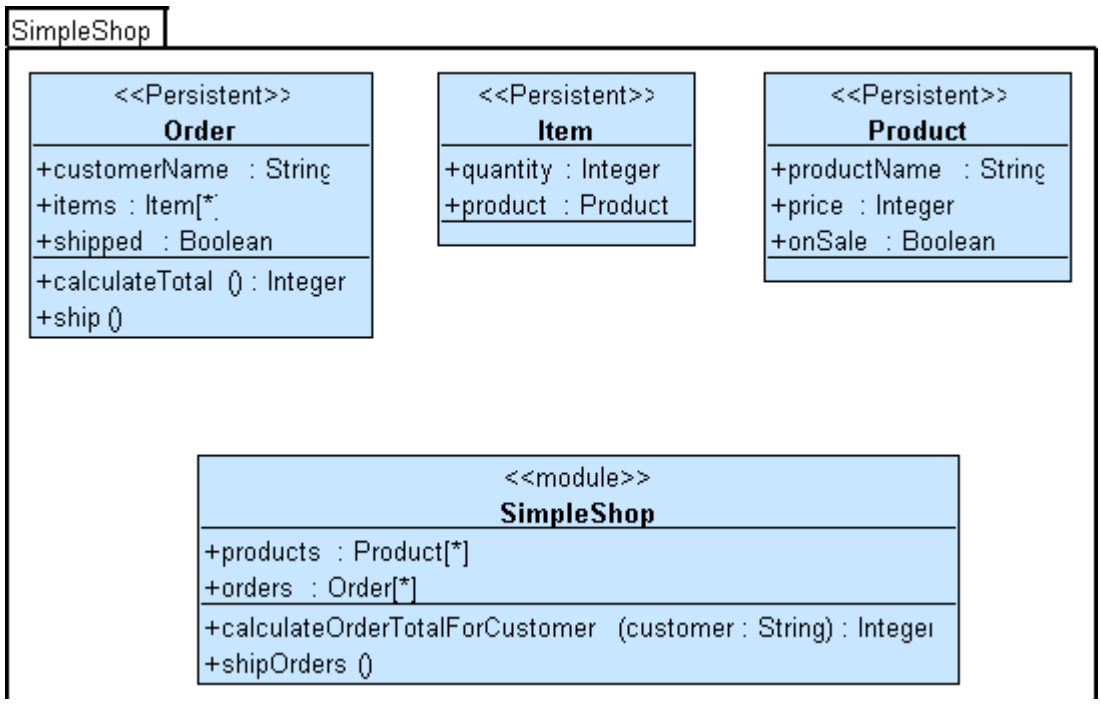

Figure 10: Provided Class Framework in VIDE

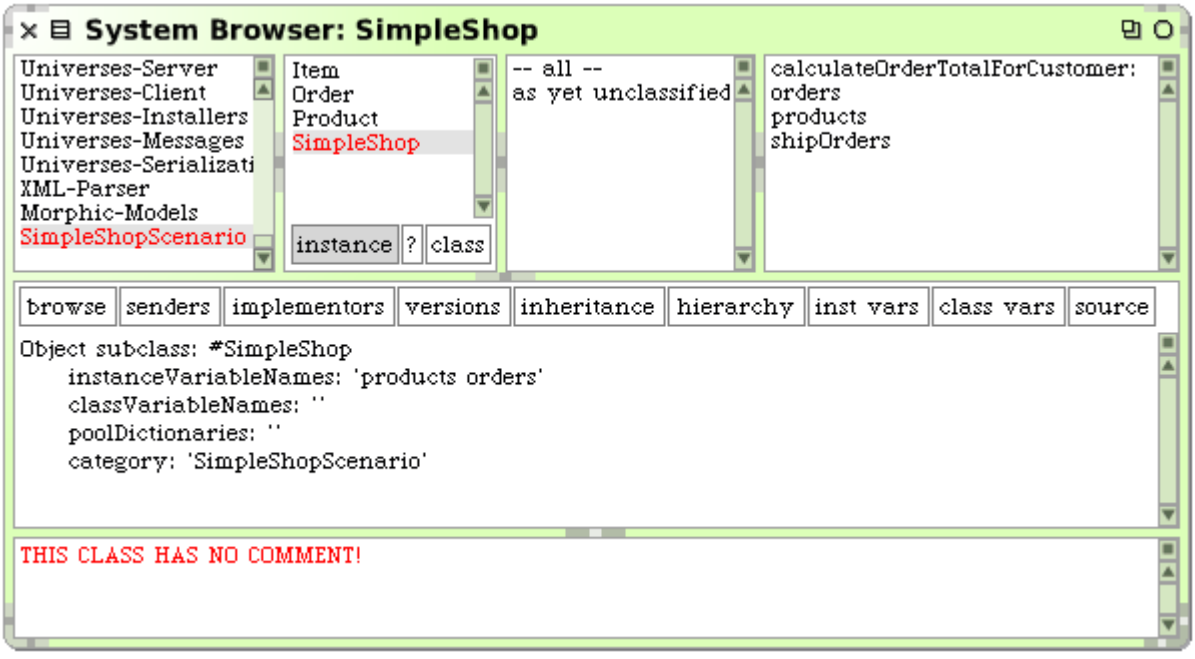

Figure 11: Provided Framework in SmallTalk

In the first task, the participants are asked to implement a method body. The participants have to iterate over a collection, picking up specific elements (using a conditional statement) and then use these elements to retrieve partial totals from them (using an operation call). After that, the partial totals have to be accumulated in a total variable, which contains the value to be returned as the result of the operation.

In the second task, a conditional statement has to be inserted in the right place. The third task requires the identification and then the removal of existing code to adapt the method behaviour as described in Task 3.

3.2.2.4 Time Measurement and Correctness Criteria

All participants were given unlimited amount of time to complete each task. The time measurement was started as soon as the participant started working on the task and stopped as soon as the participant was certain that the task is completed.

There was only one condition for the solution: For Smalltalk, the program had to compile and for VIDE, all fields of all actions had to be filled out (text input is compiled immediately). As a consequence of this condition, there were no syntax errors in the programs created by the participants.

To collect information about the kinds of syntax errors the participants made, they were observed and video-taped while solving the tasks.

3.2.3 Workshop Results

The time needed for solving the tasks was measured and we also video-taped the participants when doing that. In addition, the solution files in VIDE and in Smalltalk were saved so that we can check the semantic correctness of the solution. Syntactic errors did not exist because we set the requirement that the solution had to compile. Moreover, the participants impressions on which language was easier to learn and use were not considered as self-reported data is not objective.

The figure below shows the mean time needed for solving the tasks in both languages.

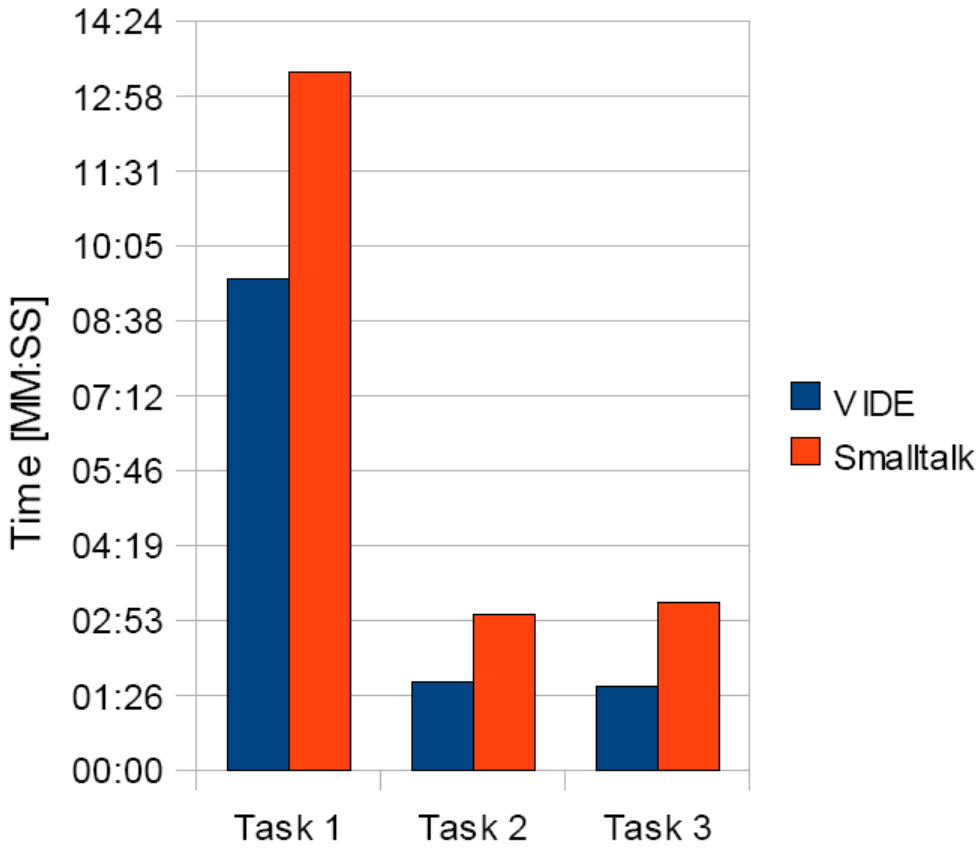

Figure 12: Mean time for solving each task in each language

Table 4 shows the time needed for solving the tasks in Smalltalk and using Squeak whereas Table 5 shows the time needed for solving these tasks in VIDE visual notation and using the respective editor. Under minor errors we refer to either a forgotten variable initialization in T1 or a semantically correct solution in Task3 without having removed all unneeded code in Task 3.

Table 4: Time needed for solving the tasks in Smalltalk

Participant	Task 1	Task 2	Task 3	Total [s]	Correctness
P1	00:20:15	00:03:37	00:03:18	1630	2 errors in T1
P2	00:16:37	00:01:33	00:02:13	1223	correct
P3	00:06:47	00:03:29	00:05:25	941	correct
P4	00:14:00	00:04:00	00:04:00	1320	correct
P5	00:15:08	00:03:32	00:02:10	1250	correct
P6	00:08:36	00:03:11	00:02:42	869	correct
P7	00:13:40	00:02:09	00:04:41	1230	1 minor error in T1
P8	00:21:01	00:05:16	00:05:09	1886	1 minor error in T1
P9	00:12:42	00:04:06	00:03:11	1199	correct
P10	00:20:29	00:02:34	00:04:57	1680	correct
P11	00:11:46	00:02:59	00:00:40	925	2 errors in T1, T3 not solved
P12	00:05:38	00:01:12	00:00:50	460	correct
P13	00:05:57	00:01:04	00:00:34	455	correct
P14	00:17:34	00:02:59	00:05:53	1586	1 error in T1
P15	00:08:41	00:03:05	00:02:29	855	1 error in each task
P16	00:15:32	00:03:12	00:03:21	1325	2 errors in T1
Mean	**00:13:24**	**00:03:00**	**00:03:13**	**1177,125**	

Table 5: Time needed for solving the tasks in VIDE Visual Notation

Participant	Task 1	Task 2	Task 3	Total [s]	Correctness
P1	00:12:26	00:01:17	00:02:09	952	3 errors in T1
P2	00:14:37	00:01:36	00:02:30	1123	correct
P3	00:06:49	00:01:20	00:02:17	626	correct
P4	00:13:00	00:02:00	00:01:00	960	correct
P5	00:11:40	00:02:17	00:01:11	908	correct
P6	00:06:22	00:01:35	00:00:40	517	correct
P7	00:10:06	00:02:21	00:01:51	858	minor error T1/error in T2
P8	00:17:27	00:02:02	00:01:17	1246	minor error in T1
P9	00:11:06	00:02:12	00:01:12	870	correct
P10	00:05:37	00:02:49	00:01:30	596	minor error in T1
P11	00:08:48	00:01:15	00:00:58	661	minor error in T3
P12	00:04:55	00:01:17	00:01:12	444	correct

P13	00:04:40	00:01:01	00:01:51	452	correct
P14	00:08:13	00:01:33	00:02:48	754	correct
P15	00:04:05	00:00:59	00:00:15	319	minor error in T3
P16	00:11:00	00:01:17	00:03:13	930	2 errors in T1
Mean	**00:09:26**	**00:01:41**	**00:01:37**	**763,5**	

3.2.4 Discussion

The mean time data shown in Table 5 shows that all three tasks were solved more quickly with the visual notation. It also shows to what extent the solution is correct. Note that no significant differences between the quality of the solutions generated with the visual language and Smalltalk were found.

To ensure that the observed results are not due to mere chance, a paired, two-tailed Student´ s t-test was used as a significance test. The so-called null hypothesis (i.e. the performances of both notations do not significantly differ) could be rejected at the 0.1% level for task 1 and 2 (with p1 = 0.0009 and p2 = 0.0002) and at the 1% level for task 3 (with p3 =0.0012). Hence, the differences are statistically very highly significant and highly significant, respectively.

The statistical significance supports the trend that was observed during the study: Participants had fewer difficulties with the VIDE visual notation than with the syntax of Smalltalk. In fact, the visual notation helped to reduce syntax hassle, resulting in significantly lower times needed to generate the solutions. We observed that the visual notation of VIDE mainly prevents several types of error such as the wrong usage of blocks, forgetting termination of statements.

There were a number of factors that could have influenced and biased the results. Although such factors were minimized, not all could be entirely prevented. Presumably the two main factors were the knowledge of similar languages, and other programming environments with features that help in programming such as auto-completion and automatic syntax formatting (which was used for Smalltalk in Squeak).

This data shows clearly that VIDE visual notation was easier to learn for the participants than a new textual programming language. Thus, this data verifies **Measurable Result 5** to a large extent **as the participants were able to program simple business applications after a short training.**

In addition, this data shows to a certain extent **a gain in the application development time that verifies Measurable Result 2, as the basic development tasks of this study were done faster with VIDE than with a traditional programming language and its IDE.** In average, there is a gain of **30%** in the development time needed for Task1, a gain of **44%** in the time needed for Task 2, and a gain of **50%** in the time needed for Task3.

3.3 pre-CIM evaluation workshop (BU)

One of the motivations for VIDE is to provide more accessible models for end-users, who may not share the modelling perspective and experience of software engineers. To this end, VIDE has suggested a sequential modelling approach, which allows users to move, via guided transformations, from simple models through the various MDA stages.

As part of the tool set, we proposed what were termed pre-CIM notations.

This deliverable reports upon experiment to assess whether our proposed pre-CIM models do, indeed, provide a more accessible or palatable starting point for users, and also, as an additional research question, attempts to gauge the impact on resultant quality of software models.

In brief our results suggest that the pre-CIM model does enhance the modeller's experience (with subjects reporting that comparable tasks were easier with the additional notation, than a control group without.

3.3.1 Introduction

The VIDE project attempts to improve the involvement of a variety of stakeholders within early, particularly CIM, modelling phases by providing an accessible modelling toolset. In particular, one of the findings of our work to date is that existing, even CIM, notations appear to be a barrier to many who do not share a software modelling background. Therefore, simple accessible notations have been developed (as part of the tool), which via guided transformations, are intended to provide a more palatable introduction to the modelling.

The intention, therefore, of our study (this evaluation) is to ascertain the impact of such (we have termed pre-CIM) notations. Thus, the following describes a number of distinct aspects of the evaluation undertaken, including an outline of the experiments undertaken, their rationale, processes and protocols, analyses of the results obtained, and general discussion of the implications of these results.

In brief, in order to assess the value of the pre-CIM notation, as a crucial part of the VIDE tool chain, two separate workshops were designed, each assessing the utility of the notations devised.

Each workshop consisted of a number of separate experiments that aimed at investigating each stage of the pre-CIM notation process. Indeed, prior to conducting the workshops much research and planning was conducted to ensure the most effective evaluation could be written. This involved researching papers on evaluating techniques and likewise topics including prior case tool evaluations. After establishing the structure of the workshops and developing the experiment documentation, a dry run was conducted in order to minimize disruptions during the live run of the workshop.

The remainder of this document is organised as follows. The conducted survey of the previously reported approaches to the tool chain evaluation is briefly presented in Section 3.3.2. Section 3.3.3 presents the summary of the designed workshops including detailed description of each stage and the applied protocol. The results of the experiment are discussed in Section 3.3.4, and analysis of the data (including descriptive statistics and hypotheses tests) are given in section 3.3.5. Finally, conclusions can be found in Section 3.3.6.

3.3.2 Research

A great deal of research was conducted prior to considering any aspect of the evaluation. Although there is much work within the general domain of empirical software engineering, and significant experience of empirical studies within both the BU research group, and the wider consortium, we were keen to revisit the literature in order to provide an experimental framework which would allow us effectively, fairly and thoroughly evaluate the Pre-CIM notation. In addition, we wished to provide a clear and transparent set of experimental guidance for experimenters, and clear instructions for subjects, typically volunteers.

In drawing up our methods and procedures, we draw heavily on two particularly pertinent and relevant sources. Al-Khilidar et al (2007) highlights a format for conducting an evaluation

which, whilst still retaining sufficient rigour, gives a clear structure sets out the essential requirements of such studies in a clear and concise fashion. Budgen and Thomson (2003) expand upon these ideas, providing detailed descriptions of the process that they adopted in evaluating a particular CASE tool. For example they highlight the importance of creating suitable experimental protocols, something which would prove particularly valuable to our workshop, since it had to be conducted in different locations.

3.3.2.1 Propositions

One of the themes of the VIDE project is that of attempting to increase the involvement of the business or client stakeholder in the requirements and specification (typically CIM), phase of development. Such increased involvement will lead to increased alignment of business needs and IT. However, it has become apparent that the models available, whilst they might offer suitable notational capability, may not be accessible to such an audience, who are often not IT specialists. Hence, the VIDE tool provides what is termed a pre-CIM phase of modelling, which, it is conjectured, lowers the barrier to entry into systems development by allowing unskilled individuals with limited or no previous modelling experience to not only participate in the systems development process, but to drive it.

To do so the notation would need to be intuitive, accessible and simple. With these considerations in mind, there are two main research questions the study sought to answer.

1) *Whether the VIDE Pre-CIM notation provides a more palatable (less onerous) route into systems development?*
2) *Whether the use of the pre-CIM notation has a positive impact on quality*?

These research questions are reflected in our research design, the nature of the data collected, our hypotheses and resulting analyses.

3.3.3 Workshop Planning

There were many areas that required planning before the workshops. An overall plan was defined from the start which helped provide guidance. This proved successful, particularly when the workshop had to be rescheduled for a week later. The ambitious deadlines allowed for a considerable headway.

The location and structure of the workshops also required careful planning. In a best case scenario over 200 students were to be available during the workshop, of which we had to split and take one group to an alternative location. This was arranged far in advanced as a number of classrooms were booked because no other rooms that could cater for that number of students were available. This was considered in the start of the protocol, covered later in Section 3.3.3.2. Another important part of the planning and protocol was the sequence in which each group were to conduct the separate experiments as this would support our evaluation. It was decided to conduct each workshop experiment sequence as shown in Table 6.

Table 6: Sequence of experiments execution for both groups of participants

	VCLL	Scrap	Analysis Palette	VCLL
Group 1	TID4 then TID6	TID2	TID3 then TID5	

Group 2		TID1	TID3 then TID5	TID4 than TID6

With this sequence the value of the sequence of pre-CIM through to CIM can be assessed by comparing group 2 who would conduct the experiments in this order against group 1 who would not. The following Section 3.3.3.1 addresses the detailed description of the developed sequence of the experiments, whereas the issues related to the protocol development are highlighted in Section 3.3.3.2.

3.3.3.1 Experiments

It was decided to conduct five experiments, four of which were essential and one optional, if there was enough time in the workshop. Because our hypothesis was to prove that the pre-CIM notation lowers the barrier to entry it was decided to run a number of separate experiments to test different stages of using the notation on two groups and to make comparisons of the results. Each experiment was based upon a cross selling opportunity scenario, with each group conducting the experiments in different sequences. To improve the credibility of each group's results it was decided to have the students work in pairs. This decision was based upon past experiences, but does to some extent seem to be supported by Al-Kilidar et al (2005). The detailed description of each of the conducted experiments is presented further in Sections - 42 -3.3.3.1.1 - 3.3.3.1.5.

3.3.3.1.1 TID1 and TID2

Group 1 was given TID2 and group 2 TID1. TID1 involved participants modelling a scenario using the bloop notation to identify Roles, Activities and Data Objects. TID2 involved participants identifying Roles, Activities and Data Objects using their own modelling technique. The aim of this experiment was to show the value and intuitiveness in the bloop notation to help identify correct Roles, Activities and Data Objects.

3.3.3.1.2 TID3

Each group was given TID3. This experiment involved completing a partial analysis palette model of the scenario. Again this was to build upon the intuitiveness of the notation, but this time at the later analysis palette stage. It would also support the value of the bloop notation and subsequent pre-CIM stages in the desired pre-CIM sequence.

3.3.3.1.3 TID4

Each group was given TID4. This experiment involved completing a partial VCLL model of the scenario. This was to support the particular sequence from pre-CIM to CIM and for comparison with TID3. The results from this experiment would clearly show whether there was value in the pre-CIM notation in building a pre-CIM model before a CIM.

3.3.3.1.4 TID5

Each group was given TID5. This experiment involved modelling an additional requirement onto a pre-drawn analysis palette model of the scenario. The aim of this experiment was to further explore the possibility that not only is the pre-CIM notation intuitive to understand, but also to adapt.

3.3.3.1.5 TID6

It was optional for each group to do this experiment depending on time. This experiment was similar to TID5 but made use of VCLL models instead of analysis palette. Again an additional requirement was suggested and a pre-drawn solution was to be selected as the answer. This experiment could have been used to further argue the whole process from starting from bloop notation and progressing sequentially helps users to understand deeper the modelling issues at the later stages of system development at the CIM level. Unfortunately, due to the limited time scale of the workshops, this experiment was not included in the live run.

3.3.3.2 Protocol

Budgen and Thomson (2003) suggested a protocol should be written so the experiment could be replicated. This was particularly important with our workshop which was to be conducted in two locations, and the sequence in which experiments were to be completed was vital to the evaluation. Much positive feedback was given from those helping us when conducting the experiments. There were however areas that could have been improved. Budgen and Thomson identified the need for a protocol to cover the process from start to finish including the analysis of data and write up. Time constraints meant we could not consider this which has resulted in separate analysis of the data. It should be noted the importance in including the gathering and analysis of the data procedures after the workshops to avoid replication.

3.3.4 Data Collected

The experimental protocol was devised in order to minimise, and ameliorate, significant threats to validity. For example, in order to ensure that assignment to groups did not introduce any bias, a questionnaire was given to all participants. Both groups reported comparable levels of modelling experience in the questionnaires provided. Therefore, it a straightforward comparison of the results achieved by both groups will not add bias to the conclusions made.

The evaluation of the collected experimental data was carried out from two different perspectives.

Firstly, the overall performance of each participant for each experiment was assessed. Following that, the data was analysed regarding the discrimination in handling each element of the TID1/TID2 and TID3 between two groups of users. TID4 and TID5 were excluded from the second part of the evaluation, as no additional discrepancy in the interpretation of results was possible due to the straightforward binary system of assessment (correct/incorrect) for these exercises.

Comparing the averaged accuracy of the participants from both groups for each stage of the workshop (see Figure 13), it can be seen that group 2, who were using the pre-CIM notation in sequence, show more accurate results than group 1. When producing this estimation the following practical constraints were introduced:

- When assessing the results of TID1/TID2, TID3 and TID4 the answer was considered correct, if no or one mistake was done when identifying correct keywords and assigning object type.
- When assessing the results of TID3, only answers containing the identification of the objects type were considered.

This estimation gives an overview of the participants in each group who produced correct answers for each task.

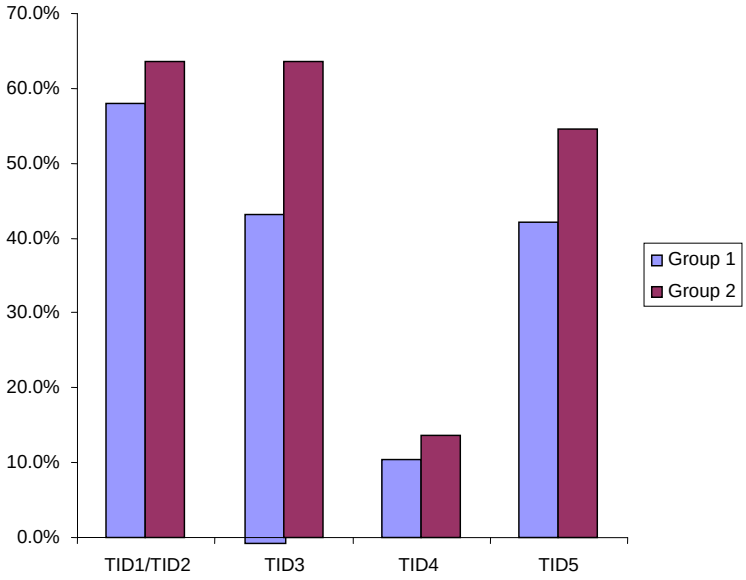

Figure 13: Accuracy of answers for both groups

Preliminarily, it can be stated that the fact that group 2 outperformed group 1 in all exercises (with an overall performance of each participant for each experiment assessed) supports our hypothesis. Interestingly, group 1 also found the experiments slightly harder, which is analogous to, and suggestive of added value in the consequent execution of models from pre-CIM or 'bloop' analysis model - to Analysis Palette model – through to VCLL model. This is shown in Figure 14.

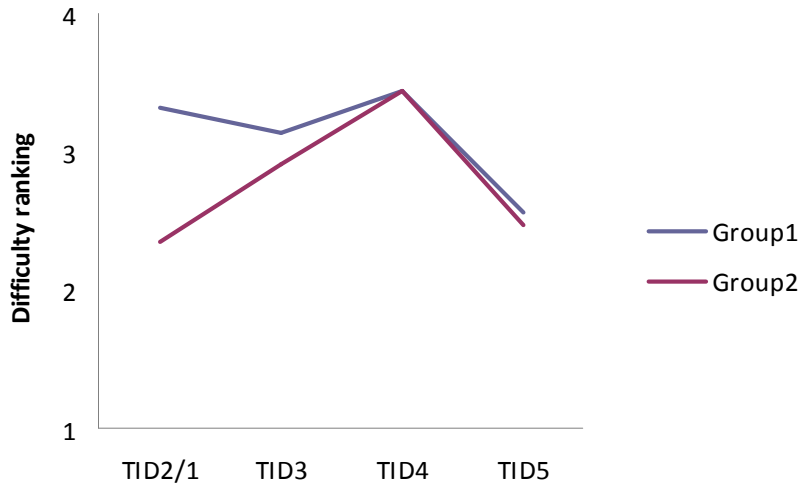

Figure 14: Comparison of difficulty rankings averaged by group, estimated by each participant for each exercise, with 1 standing for the easiest and 5 for the hardest

Moreover, the detailed analysis shows that the answers produced by Group 2 are characterised not only with better accuracy on average, by with smaller rate of standard deviation (see Table 7 and

Table 8).

Table 7: Comparison of performance of two groups at pre-CIM level of modelling (TID1/TID2). Group 2 uses bloops notation.

Keywords	*Group 1 (total 39 pairs, TID2)*					*Group 2 (total 23 pairs, TID1)*				
	Role	Activity	Data Object	Not Relevant	%Correct	Role	Activity	Data Object	Not Relevant	%Correct
Customer	36	1	2	0	92.31%	23	0	0	0	100.00%
Staff Member	32	1	1	5	82.05%	13	0	0	10	56.52%
Book Keeper	25	3	4	6	15.79%	4	0	0	19	82.61%
Personalised Homepage	2	4	21	12	53.85%	0	2	19	2	82.61%
Enter Order	0	37	2	0	94.87%	1	22	0	0	95.65%
Confirm Order	0	37	2	0	94.87%	1	20	2	0	86.96%
Order Dispatched	1	32	4	2	82.05%	0	18	1	4	78.26%
Administrator	25	2	1	11	92.31%	9	0	0	14	100.00%
Email Confirmation	0	21	17	1	97.44%	0	11	12	0	100.00%
Update Stock	2	26	11	0	94.87%	0	18	5	0	100.00%
Payment Details	0	6	33	0	84.62%	0	2	20	1	86.96%
Item Selected	0	15	19	5	87.18%	1	11	11	0	95.65%
Log In	3	33	2	1	84.62%	1	21	0	1	91.30%
List Potential Products	1	14	22	2	92.31%	1	5	16	1	91.30%
Order Received	0	15	19	5	12.82%	0	12	5	6	26.09%
IP Address	1	1	8	29	74.36%	1	1	5	16	69.57%
Average accuracy					**77.27%**					**83.97%**
Standard deviation					**26.78%**					**19.62%**

**Green indicates what we perceived as correct results.*

Table 8: Comparison of performance of two groups at the Analysis Palette level of modelling (TID3)

Keywords	*Group 1 (total 23 pairs)*				*Group 2 (total 11 pairs)*			
	Role	Activity	Data object	n/a	Role	Activity	Data object	n/a
Order Confirmation	-	17.4%		-	-	27.3%		9.0%
Check Product Availability	-	30.4%	-	-	-	45.4%	9.0%	-
Staff	8.7%	-	-	-	9.0%	-	-	-
Customer	100.0%	-	-	-	100.0%	-	-	-
Payment Details	-	17.4%	60.9%	-	-	8.7%	27.3%	-
Book Keeper	13.0%	-	-	-	9.0%	-	-	-
Cancel Order	-	26.1%	-	-	-	36.4%	-	-
Dispatch Order	-	78.3%	8.7%	-	-	81.8%	-	-
Save Order	-	26.1%	4.3%	4.3%	-	9.0%	9.0%	-
Product	-	4.3%	56.5%	4.3%	-	-	72.7%	-
Log Out	-	26.1%	-	4.3%	-	36.4%	-	-

**Green indicates what we perceived as correct results*

3.3.5 Data Analysis

There were two fundamental research issues that this study sought to address:

1) *Whether the VIDE Pre-CIM notation provides a more palatable (less onerous) route into systems development?*
2) *Whether the use of the pre-CIM notation has a positive impact on quality*?

These concepts are reflected in the choice of variables (data collected) and resulting hypotheses. Notably, we attempt to gauge quality by examination of direct attributes of the models created by the subjects, in both a positive and negative manner, that is, we consider the correct identification of objects, mistakes made in identifying object types, the correct identification of keywords, and mistakes made in identifying irrelevant keywords. Hypotheses 1, 2, 4, 5, 6, and 7 (given below) relate to this research question (on resulting model quality).

The barrier to entry is clearly more subjective, but we attempt to assess this aspect by asking participants to score the difficulty of the tasks they have undertaken. Clearly, this is a measure of the perception of difficulty rather than an absolute measure of difficulty. However, we believed that this reflects the reality of getting clients involved in the CIM phases of development. If they perceive notations to be difficult or onerous, they will not engage, whereas, in contrast, if they perceive the tasks as within their capabilities and experience they will more readily contribute.

3.3.5.1 Hypotheses

H1: Group 1, using the pre-CIM notation, will make fewer mistakes in identifying object types (in TID1/2) than the equivalent group without.

H1A: There will be no significant difference in the amount of mistakes made in either group.

H2: Group 1, using the pre-CIM notation, will make fewer mistakes in identifying irrelevant keywords (in TID1/2) than the equivalent group without.

H2A: There will be no significant difference in the amount of mistakes made in either group.

H3: Group 1, using the pre-CIM notation perceived (rated) the tasks (TID1/2) more difficult than as easier (lower difficulty score) than group 2.

H3A: Both groups perceived the tasks as equally difficult.

H4: Group 1, using the pre-CIM notation, will identify more correct keywords (in TID3) than the equivalent group without.

H4A: There will be no significant difference in the number of keywords identified correctly by either group.

H5: Group 1, using the pre-CIM notation, will identify more correct object types (in TID3) than the equivalent group without.

The alternative hypothesis being

H5A: There will be no significant difference in the number of object types identified correctly by either group.

H6: The group with pre-CIM will perform better at task five than the group without

H6A: There will be no difference in the performance of groups on this task

H7: Group 1, using the pre-CIM notation, will identify more correct keywords (in TID4) than the equivalent group without.

The alternative hypothesis being

H7A: There will be no significant difference in the number of keywords identified correctly by either group.

3.3.5.1.1 Description of Data

For each of the hypotheses above, for which we have numerical data, this is depicted as a standard box plot. That is the results from the group with pre-CIM notation are represented alongside the group without.

The box itself represents the inter-quartile range, that is, that portion of the data between the 25th and 75th percentile, with the whiskers being the lower and upper values that are not outliers or extreme values. The bold line across the box represents the median.

Hence, these plots give a simple visual representation of the relative performance of each group, for each hypothesis, which is then supplemented by simple statistics and the formal hypothesis test.

3.3.5.1.2 Statistical Tests Undertaken

Hypotheses tests were carried out using independent samples t-tests, utilising SPSS version 16.

In each case a pre-test of group statistics shows the number from each group (missing data items for each variable being omitted for individual tests rather than across all tests, in order to provide as much data as possible for each test). In addition, in each case a Table is given showing:

- Levene's Test for Equality of Variances,
- the t-value,
- the calculated degrees of freedom,
- and the significance or p-value for a two tailed test.

Based on our propositions we had taken the positive step of forming single tailed hypotheses. Our tables of tests, generated by SPSS, however, give a significance assuming two-tailed tests, therefore, we refer to the equivalent single tailed value (half of the table figure for a 5% test of significance) in the accompanying explanatory text for each hypothesis.

Rather than explain the data provided in generic terms, analysis of hypothesis one provides further commentary on the statistical assumptions (for example, tests of variance), and further hypotheses tests, allowing remaining tests, by adopting the same results format, to provide more concise commentary.

3.3.5.2 Mistakes in Identifying Object Types

Recall that, in order to judge whether the pre-CIM notations have a positive impact on the quality of models, we wish to measure these quality attributes both in terms of mistakes made and items correct. One of these measures of quality is to consider the number of mistakes made in identifying objects (in the comparable tasks TID1 and TID2).

Our formal hypothesis is that:

H1: Group 1, using the pre-CIM notation, will make fewer mistakes in identifying object types (in TID1/2) than the equivalent group without.

The alternative hypothesis being

H1A: There will be no significant difference in the amount of mistakes made in either group.

The box plot (Figure 15: Box plots for Hypothesis 1) shows the performance of both groups side by side. Note that although the medians appear similar the group without pre CIM models clearly appears to have far more pairs making higher numbers of mistakes.

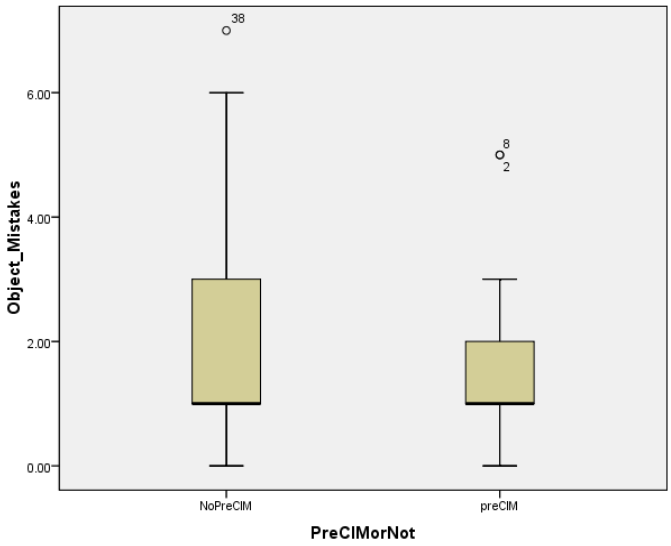

Figure 15: Box plots for Hypothesis 1

This is also suggested by the mean and related descriptive statistics for this data (Table 9).

Table 9: Descriptive Statistics for Hypothesis 1

Group Statistics

	Pair_ID	N	Mean	Std. Deviation	Std. Error Mean
Object_Mistakes	>= 2.00	23	1.5217	1.41001	.29401
	< 2.00	39	1.9487	1.91873	.30724

The most important statistic to note (Table 9), is that the means of the two groups appear to be different, that is the second group, without the pre-CIM notation have a mean error rate of 1.95 (3 s.f.), compared to only 1.52 (3 s.f.) for the group with the pre-CIM notation.

However, though these results are suggestive of the treatment (pre-CIM) leading to improvements in model quality, in order to examine this hypothesis (and others to follow) we undertake an independent samples t-test.

Although standard deviation appears slightly different, as one might expect, this does not mean that we need to assume a different underlying variance. Indeed, Table 10: Tests for Hypothesis 1, presents the results of a number of statistical tests (including the main independent samples t-test), for both equal variance assumed and not assumed, on the same data. Importantly, Levene's statistic has a p-value of .099, greater than 0.05, which means that we can assume equal variances. (Further hypotheses will also include this data and assume equal variances unless there is evidence to the contrary).

The t-value, with a calculated degrees of freedom of 60, is -.928. The two tailed significance value for this data is .0357, therefore a one tailed value would be half of this (or 0.1785).

Hence, despite the appearance, from both box-plots and means, that the pre-CIM group performed better, the test does not show a significant difference between these means, and we must accept the null hypothesis. That is, although there does appear to be some improvement, that is fewer mistakes were made when using the pre-CIM notation, the difference is not statistically significant.

Table 10: Tests for Hypothesis 1

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Object_Mistakes	Equal variances assumed	2.816	.099	-.928	60	.357	-.42698	.45994	-1.34700	.49304
	Equal variances not assumed			-1.004	56.960	.320	-.42698	.42525	-1.27854	.42459

In summary, examination of means and of the box plots suggests that the group using the pre-CIM notation does indeed make fewer mistakes in identifying object types, but this improvement is not sufficient to be statistically significant.

3.3.5.3 Keyword Mistakes (Irrelevant Keywords)

A similar measure of quality is to consider the number of mistakes made in identifying keywords.

Our hypothesis is that

H2: Group 1, using the pre-CIM notation, will make fewer mistakes in identifying irrelevant keywords (in TID1/2) than the equivalent group without.

The alternative hypothesis being

H2A: There will be no significant difference in the amount of mistakes made in either group.

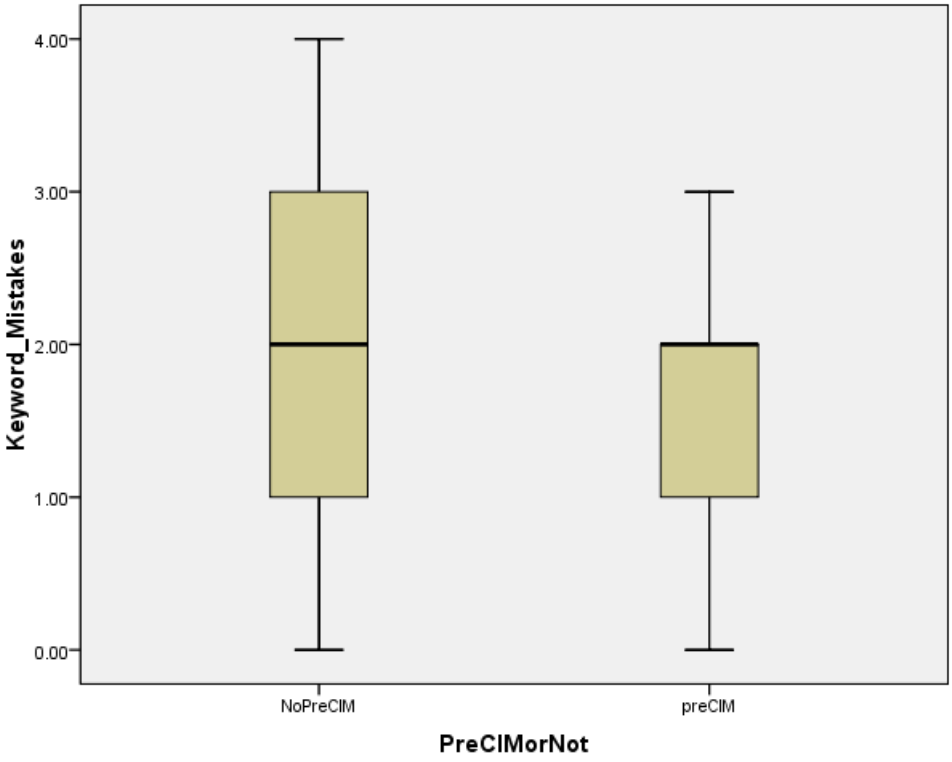

Figure 16: Box plots for Hypothesis 2

The figure above appear to suggest that the group without pre-COM tend to make more mistakes than the group who used the pre-CIM models.

Table 11: Descriptive Statistics for Hypothesis 2

Group Statistics

	Pair_ID	N	Mean	Std. Deviation	Std. Error Mean
Keyword_Mistakes	>= 2.00	23	1.6087	1.03305	.21541
	< 2.00	39	1.9487	1.12270	.17978

This is borne out by the descriptive statistics, see Table 11, in that the second (no pre CIM) group has a higher mean, for the amount of mistakes made in correctly identifying keywords.

For the hypothesis test, Table 12, the t-value, with calculated degrees of freedom of 60, is -.1.186. The two tailed significance value for this data is 0.240, therefore a one tailed value would be half of this (or 0.12). In other words there is a p value of approx 12%. Hence, although this is suggestive of an improvement it is not statistically significant.

Table 12: Tests for Hypothesis 2

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Keyword_Mistakes	Equal variances assumed	.003	.955	-1.186	60	.240	-.34002	.28675	-.91360	.23356
	Equal variances not assumed			-1.212	49.436	.231	-.34002	.28057	-.90372	.22368

3.3.5.4 Difficulty of Task (T1&T2)

The previous two hypotheses related to direct measures of the quality of models, by examination of errors or mistakes made. In contrast, the following hypotheses relates to the perceived difficulty of the previous tasks.

H3: Group 1, using the pre-CIM notation perceived (rated) the tasks (TID1/2) more difficult than as easier (lower difficulty score) than group 2.

The alternative hypothesis being

H3A: Both groups perceived the tasks as equally difficult.

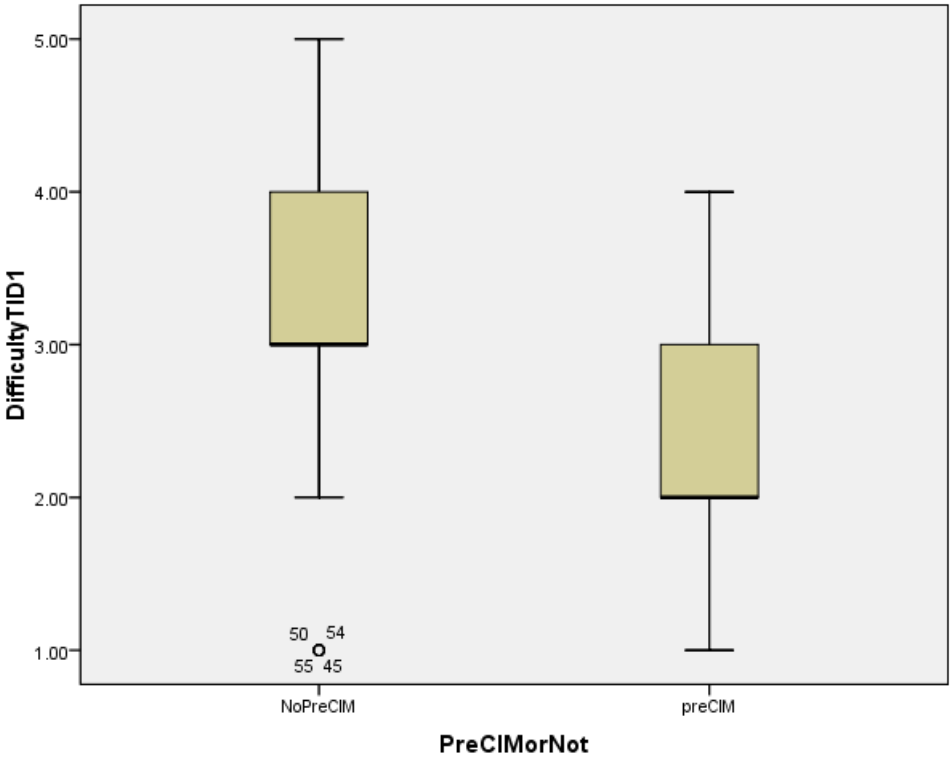

Figure 17: Box plots for Hypothesis 3

The box plot (Figure 17) shows that the group with pre CIM appear to do much better. Indeed, the median for the group without pre-CIM is at a comparable error rate to the top of the box (75^{th} percentile) for the group with pre-CIM, and there is little overlap in inter-quartile range.

Table 13: Descriptive Statistics for Hypothesis 3

Group Statistics

	Pair_ID	N	Mean	Std. Deviation	Std. Error Mean
DifficultyTID1	>= 2.00	20	2.3500	.93330	.20869
	< 2.00	38	3.3158	1.21043	.19636

This picture is also borne out by the descriptive statistics (Table 13), where, from examination of mean scores, it appears that the pre-CIM group give a much lower mean rating for difficulty, 2.35 (3 s.f.) than the other group, 3.32 (3 s.f.)

Table 14: Tests for Hypothesis 3

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
DifficultyTID1	Equal variances assumed	1.333	.253	-3.110	56	.003	-.96579	.31053	-1.58786	-.34372
	Equal variances not assumed			-3.370	48.152	.001	-.96579	.28655	-1.54188	-.38970

Furthermore, the significance of the t-test (0.003) for 2 tailed, (0.0015 for single tailed) shows a highly significant ($P<0.01$) difference in the perceived difficulty of the tasks (see Table 14).

Hence, it is clear that the group using the pre-CIM notation perceived the task to be easier than the group without.

3.3.5.5 Number of Keywords Identified Correctly (TID3)

The following hypothesis attempts to gauge whether the pre-CIM notation has a positive impact on the quality of the model, the variable under scrutiny being the number of keywords identified correctly (from task 3).

H4: Group 1, using the pre-CIM notation, will identify more correct keywords (in TID3) than the equivalent group without.

The alternative hypothesis being

H4A: There will be no significant difference in the number of keywords identified correctly by either group.

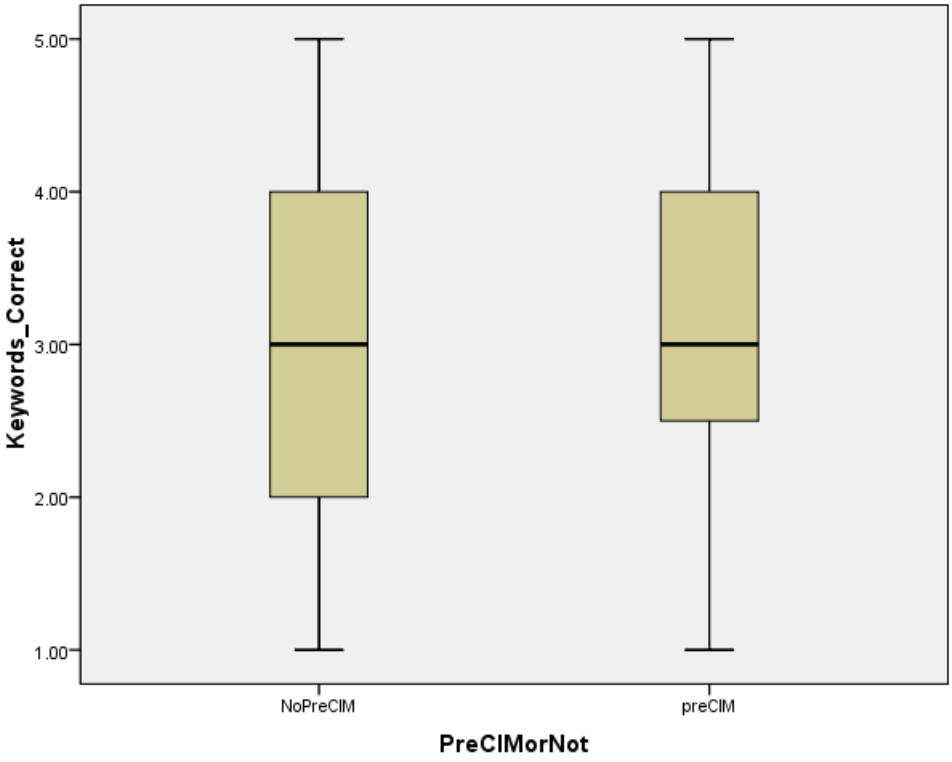

Figure 18: Box plots for Hypothesis 4

The box plot in figure above shows that both groups have a similar median, and similar values for the 75th percentile (top of the box), and upper and lower values. The main difference appears to be that the 25th percentile for the group without pre-CIM is lower, indicating that a greater proportion of the group have lower scores.

Table 15: Descriptive Statistics for Hypothesis 4

Group Statistics

	Pair_ID	N	Mean	Std. Deviation	Std. Error Mean
Keywords_Correct	>= 2.00	23	3.2174	1.24157	.25889
	< 2.00	37	2.8108	1.17468	.19312

Indeed, despite similar medians, these lower scores clearly impact the mean, see Table 15, with a mean of 3.2 correct for the pre-CIM group against only 2.8 for the other (no pre-CIM) group.

Table 16: Tests for Hypothesis 4

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Keywords_Correct	Equal variances assumed	.011	.918	1.275	58	.207	.40658	.31877	-.23150	1.04466
	Equal variances not assumed			1.259	44.816	.215	.40658	.32298	-.24401	1.05717

For the t-test, the significance of 0.207 equates to a single tailed p value equivalent to 0.1035 (see Table 16). Although this is not significant at the normal 5% level, and we must accept the null hypothesis, it is extremely close to the 10% level which is sometimes seen as a less rigorous indicative level of significance.

3.3.5.6 Number of Correct Object Types Identified (TID3)

The following hypothesis attempts to gauge whether the pre-CIM notation has a positive impact on the quality of the model, the variable under scrutiny being the number of object types identified correctly.

H5: Group 1, using the pre-CIM notation, will identify more correct object types (in TID3) than the equivalent group without.

The alternative hypothesis being

H5A: There will be no significant difference in the number of object types identified correctly by either group.

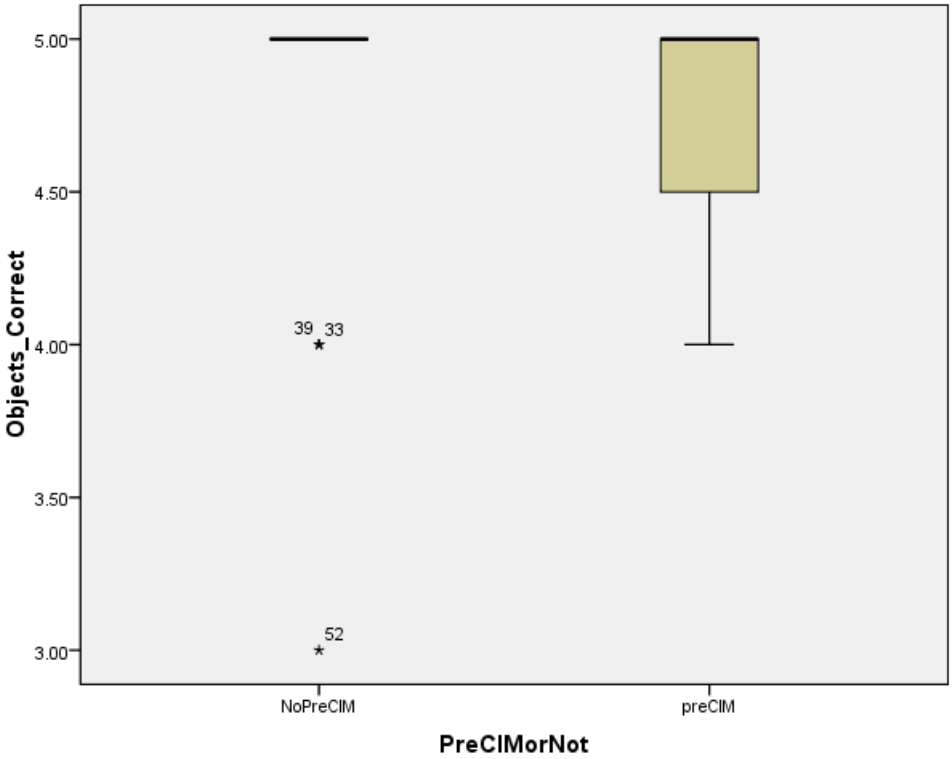

Figure 19: Box plots for Hypothesis 5

The box plot (Figure 19: Box plots for Hypothesis 5) shows an extremely unusual pattern, in that, for both groups, the median is also the maximum value. This is explained by the fact that, of the data present, many of the subjects identified correctly the maximum 5 of 5 objects, and only very few only 3.

Table 17: Descriptive Statistics for Hypothesis 5

Group Statistics

	Pair_ID	N	Mean	Std. Deviation	Std. Error Mean
Objects_Correct	>= 2.00	11	4.7273	.46710	.14084
	< 2.00	22	4.7727	.52841	.11266

Hence, the means are also very similar (see Table 17), and the standard deviation relatively small for both groups. This is unusual in our analysis so far, and, even without recourse to statistical analysis it is clear that there is little difference in the performance of each group. Nevertheless, for completeness the t-test statistics are computed below (Table 18).

Table 18: Tests for Hypothesis 5

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Objects_Correct	Equal variances assumed	.043	.838	-.242	31	.811	-.04545	.18812	-.42914	.33823
	Equal variances not assumed			-.252	22.504	.803	-.04545	.18035	-.41899	.32809

Clearly, there is no significant difference in the results from each group, and thus it does not appear that the use of the pre-CIM notation leads to any improvement in the number of object types identified.

3.3.5.7 Overall model considered correct / incorrect

H6: The group with pre-CIM will perform better at task five than the group without

H6A: There will be no difference in the performance of groups on this task

Task five is judged as either correct or incorrect, rather than having a score. Hence, it is difficult to assess the significance of any result. However, 54.54% from the group with pre-CIM were considered correct against 42.1 for the group without.

3.3.5.8 Number of Keywords Identified Correctly (TID4)

H7: Group 1, using the pre-CIM notation, will identify more correct keywords (in TID4) than the equivalent group without.

The alternative hypothesis being

H7A: There will be no significant difference in the number of keywords identified correctly by either group.

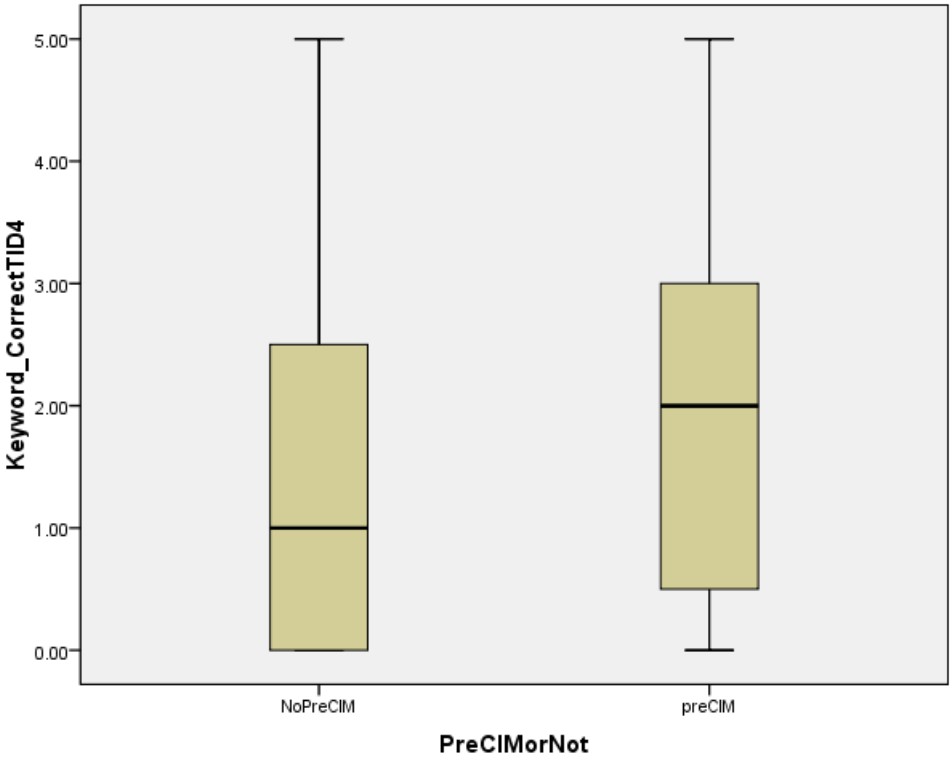

Figure 20: Box plots for Hypothesis 7

The box plots (Figure 20) show a higher median, and both 25th and 75th percentile for the pre-CIM group, once more suggesting that the group with pre-CIM models fared better, in this case correctly identifying more keywords correctly, and this is also reflected in a higher mean for the pre-CIM group.

Table 19: Descriptive Statistics for Hypothesis 7

Group Statistics

	Pair_ID	N	Mean	Std. Deviation	Std. Error Mean
Keyword_CorrectTID4	>= 2.00	23	1.7826	1.50625	.31407
	< 2.00	39	1.3846	1.54945	.24811

Once more it appears from the means that the group with pre-CIM appears to fare better, in that the mean keywords correct is 1.78 against only 1.38 for the group without pre-CIM models (Table 19). However, as Table 20 shows (with a two-tailed significance of 0.325, equating to a single tailed of 0.16) this improvement is not statistically significant.

Table 20: Tests for Hypothesis 7

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Keyword_CorrectTID4	Equal variances assumed	.008	.929	.987	60	.328	.39799	.40323	-.40859	1.20458
	Equal variances not assumed			.994	47.351	.325	.39799	.40025	-.40705	1.20304

3.3.5.9 Summary of Hypothesis Testing

H1: Group 1, using the pre-CIM notation, will make fewer mistakes in identifying object types (in TID1/2) than the equivalent group without.

H1A: There will be no significant difference in the amount of mistakes made in either group.

Group 1 perform better, but this is not statistically significant. Therefore reject H1 in favour of H1A.

H2: Group 1, using the pre-CIM notation, will make fewer mistakes in identifying irrelevant keywords (in TID1/2) than the equivalent group without.

H2A: There will be no significant difference in the amount of mistakes made in either group.

Group 1 perform better, but this is not statistically significant. Therefore reject H2 in favour of H2A.

H3: Group 1, using the pre-CIM notation perceived (rated) the tasks (TID1/2) more difficult than as easier (lower difficulty score) than group 2.

H3A: Both groups perceived the tasks as equally difficult.

Group 1 perform much better, and this improvement is highly significant. Therefore accept H3.

H4: Group 1, using the pre-CIM notation, will identify more correct keywords (in TID3) than the equivalent group without.

H4A: There will be no significant difference in the number of keywords identified correctly by either group.

Group 1 perform better, and this is on the bounds of weak significance (p of approximately 0.10). Therefore reject H4 in favour of H4A.

H5: Group 1, using the pre-CIM notation, will identify more correct object types (in TID3) than the equivalent group without.

H5A: There will be no significant difference in the number of object types identified correctly by either group.

Very similar performance from both groups; therefore accept the alternative hypothesis H5A.

H6: The group with pre-CIM will perform better at task five than the group without

H6A: There will be no difference in the performance of groups on this task

Group 1 perform better, with a higher percentage correct, but no tests of significance undertaken.

H7: Group 1, using the pre-CIM notation, will identify more correct keywords (in TID4) than the equivalent group without.

H7A: There will be no significant difference in the number of keywords identified correctly by either group.

Group 1 perform slightly better, but this is not statistically significant. Therefore reject H7 in favour of H7A.

3.3.5.10 Discussion of Findings (Analysis of Data)

For each of our hypotheses, the data shows that the group who used the pre-CIM model performed better. This can be seen from our box-plots, and from the descriptive statistics, such as the mean. However, in all cases bar one, the results were not statistically significant, with one of the results being very close to the 10% significance level.

Given, however, that in all cases there appears to be a consistent positive effect, in favour of the group using pre-CIM, our conjecture is that the study undertaken had sufficient power to be able to show the impact of the treatment (the treatment being the use of the pre-CIM notation against our baseline of no pre-CIM).

In other words, though there appears to be a positive impact on quality, this effect is not great enough to be demonstrated within this study. With hindsight, using uneven group sizes we were constrained in the statistical tests that could be applied, and coupled with the use of pairs, which reduced the number of separate subjects, the power appears insufficient to demonstrate statistical significance.

Hence, in terms of the original propositions, we cannot provide any significant statistical evidence that the use of pre-CIM models improves the quality of resulting models. Despite this, the results are consistent and promising, and suggestive of the need for further, perhaps larger, study.

The only strongly significant result is was for the hypothesis (H3) which attempts to gauge difference in the perception of difficulty. Hence, it appears that, irrespective of the fact that the models cannot be demonstrated to be of higher quality, the subjects felt that the task was easier (or at least reported that it was) when they used the pre-CIM models.

3.3.6 Conclusions

This evaluation has sought to investigate the utility of proposed VIDE modelling notations. In particular, the evaluation sets out to gauge the effect of our treatment, introducing a 'pre-CIM', notation as part of a series of models within the VIDE tool chain.

The motivation for our introduction of such notations is that current modelling approaches appear to act as a deterrent factor in the involvement of the very stakeholders who often best understand the needs and requirements for proposed systems development. Hence, by providing a more accessible entry point, we would hope to improve stakeholder involvement within the crucial CIM phases of development.

In doing so, we would hope that such models would be perceived, by the modellers as providing a modelling experience which was less onerous than the alternative. In addition, if such a modelling experience improves involvement, one would hope that it does so without any detriment to the resulting software design. Indeed, in our study questions we took the even bolder step of suggesting that the models, whilst improving the perception of the modelling phase, might also actually improve resultant software models. This provided two principal research questions: 1) Whether the VIDE Pre-CIM notation provides a more palatable (less onerous) route into systems development, and 2) Whether the use of the pre-CIM notation has a positive impact on quality?

This was investigated by comparing the performance of two comparable sets of subjects, one using our pre-CIM notation, and one without, across a series of sequential tasks (mirroring the proposed VIDE tool chain). Performance on these tasks was scored, and participants were also asked to rate their assessment of the difficulty of the tasks for both groups. Data for this performance was analysed visually, e.g., using box-plots, descriptive statistics (such as means and standard deviations) were generated, and tests of significance (principally t-tests) carried out.

In brief our findings were that the treatment group, using the pre-CIM notation, did find the tasks less difficult than the control group, and that the difference in mean difficulty score was highly significant. In addition, we found that in all other tasks the treatment (pre-CIM) group fared better, but, we conjecture due to the limited power of the tests, these did not appear statistically significant improvements.

Hence, the main motivator for introduction of such notations, reducing the deterrent of existing notations appears to be validated by this study. In addition, rather than concern for the possible negative impact of introducing such notations on software model quality our results suggest (though a larger study would be required to show this effect statistically) that model quality may also be improved.

3.4 Evaluation of VIDE-Defect Detector (IESE)

One goal of VIDE was to explore techniques for "*quality assurance, which decreases the number of quality defects and should be applied to the platform-independent level to become even more effective*" (VIDE-Annex, 2005). As a measurable result we stated that "*With the discovery of quality defects and following model refactorings the qualitative and quantitative quality as measured against a specific quality model (probably based on ISO 9126/25000 but for MDA) will increase by at least 20%.*" (VIDE-Annex, 2005).

Therefore, the overall goal of this empirical study is to evaluate the effect of the context-sensitive quality defect diagnosis approach realized in VIDE-DD on quality improvement and acceptance by users. VIDE-DD supports software modelers with the presentation of quality defects (prioritized according to a given quality model) within the model diagrams as well as a list view. The features of VIDE-DD should sensiblize the software modellers about potential problems, focus the attention to important quality defect, improve the model quality if quality defects are removed, and result in a positive reception by software modellers.

To approach this evaluation, we analyzed, measured, and compared the quality of PIM developed with and without the quality defects diagnosis approach realized within VIDE-DD. Measurement will be conducted subjective as well as objective.

3.4.1 Study Design

3.4.1.1 Hypotheses about VIDE-DD effects

Based on the features of VIDE-DD we suspect the truthfulness of the following hypotheses.

Regarding quality improvement, we state the following alternative and null hypotheses:

- H_{1a}: Using VIDE-DD has a *positive effect on the model quality* compared to the manual inspection of the software model (in the same timespan).
- H_{0a}: There is no significant effect on quality improvement.

Regarding the diagnostic efficiency, we state the following alternative and null hypotheses:

- H_{1b}: Using VIDE-DD results in *more quality defects diagnosed* compared to the manual inspection of the software model (in the same timespan).
- H_{0b}: There is no significant effect on the amount of diagnosed quality defects.

Regarding the handling efficiency (i.e., stating decisions or context factors), we state the following alternative and null hypotheses:

- H_{1c}: Using VIDE-DD results in *more quality defects handled* compared to the manual inspection of the software model (in the same timespan).
- H_{0c}: There is no significant effect on the amount of diagnosed quality defects.

Regarding acceptance and use of QDD technology, we state the following alternative and null hypotheses:

- H_{1d}: Quality defect diagnosis tools (such as VIDE-DD) have a significant *positive rating towards acceptance* and intention to be used.
- H_{0d}: There is no significant rating on the acceptance and intended use.

3.4.1.2 Arguments and Conjectures about VIDE-DD effects

Beside these hypotheses, which were evaluated in the study, we did also have the following arguments that advocate the diagnosis of quality defects on the PIM level. While no empirical evaluation will be conducted to prove these arguments we state them as conjectures:

- Conjecture 1: In a **multi-platform scenario**, quality defect diagnosis and removal on PIM level reduces the amount of work n times, as it is only required once on PIM level versus n times on PSM and/or Code level (In the envisioned multi-PSM vision of MDA with 1 PIM, n PSMs, and n Code-level systems).
- Conjecture 2: In a **single-platform scenario**, quality defect diagnosis and removal on PIM level reduces the amount of work to 50% (or 33%), as it is only required once on PIM level versus once on PIM and once on PSM level (33% if PSM and Code is used).
- Conjecture 3: Instead of checking for defects on the code level and reverse-engineering the changes up to the PIM only the PIM is changed - (no effort for reverse-engineering and supervision of changes to PIM required).
- Conjecture 4: The persistent storage of context factors and decisions within the PIM is used to block and hide mis-diagnosed or inevitable quality defects and, therefore, reduces the amount of work as the software modeller does not have to re-inspect the mis-diagnosed or inevitable defects.

3.4.1.3 Factors & Data Collection

This section states the variables considered in the analysis of the causal relationship of the effect of VIDE-DD. In order to measure the effect of VIDE-DD on the dependent factors several independent and confounding factors had to be observed as well.

3.4.1.3.1 Impact on Quality – H_{1a}

Currently, the quality of models and software systems in general is hard to determine. Several frameworks to develop quality models have been developed (Ortega et al., 2003) or standardized (ISO/IEC-9126-1, 2003). However, no concrete metrics-based model is widely accepted – most organizations construct their own ones using the frameworks. Furthermore, we do not have a tool to measure all necessary metrics necessary for some of the documented quality models. Furthermore, the effects of some quality defects, such as “Type Embedded in Name” or “Uncommunicative Name”, are not captured using model metrics.

Hence, in order to evaluate the improvement of the model quality, we collect the following data:

- The subjectively perceived quality improvement using an ISO9126-based questionnaire (for H_{1a} - *subjective*)
- The number of quality defects diagnosed (and removed or otherwise handled) used as an indicator for the improvement of ISO9126 quality aspects (for H_{1a} - *objective*)

The indirect factor used to evaluate the *subjective* part of hypothesis H_{1a} is perceived quality (PQ). For a subject i this is calculated via the following formula (based on the characteristics of ISO9126):

$$PQ_i = \frac{QM_i + QP_i + QE_i + QR_i + QF_i + QU_i}{6}$$

The indirect factor used to evaluate the *objective* part of hypothesis H_{1a} is removed quality defects (RQD), which is based upon the following factors:

- Number of quality defects removed (NR)
- Number of quality defects handled with decisions (ND)
- Number of quality defects handled with context factors (NC)

For a subject i this is calculated via the following formula (based on the characteristics of ISO9126):

$$RQD_i = NR_i + ND_i + NC_i$$

The quality characteristics maintainability (QM), portability (QP), efficiency (QE), reliability (QR), functionality (QF), and usability (QU) will be calculated from the associated sub-characteristics (see post-test questionnaire 0).

However, as the currently diagnosed quality defects do not cover all these quality characteristics we will evaluate differences in the perceived quality improvement for individual quality characteristics (e.g., only maintainability).

3.4.1.3.2 Diagnostic efficiency – H_{1b}

The efficiency of the diagnosis is based upon the following factors:

- Number of quality defects diagnosed (NQD)
- Time required for the diagnosis by the subject (DT)

The indirect factor used to evaluate the hypothesis H_{1b} is diagnosis efficiency (DE). For a subject i this is calculated via the following formula:

$$DE_i = \frac{NQD_i}{DT_i}$$

3.4.1.3.3 Handling efficiency – H_{1c}

The efficiency of the handling is based upon the following factors:

- Number of quality defects handled with decisions (ND)
- Number of quality defects handled with context factors (NC)
- Time required for the diagnosis by the subject (DT)

The indirect factor used to evaluate the hypothesis H_{1c} is handling efficiency (HE). For a subject i this is calculated via the following formula:

$$HE_i = \frac{ND_i + NC_i}{DT_i}$$

3.4.1.3.4 Acceptance and Use of VIDE-DD (UTAUT) – H_{1d}

One part of the evaluation was focused on the acceptance and potential use of VIDE-DD by the subjects. The Unified Theory of Acceptance and Use of Technology (UTAUT) (Venkatesh et al., 2003) was used as it represents the most mature instrument to measure the individual acceptance of information technology. UTAUT consists of three direct determinants of intention to use (performance expectancy, effort expectancy, and social

influence) and two direct determinants of usage behaviour (behavioural intention and facilitating conditions). This model captures the following aspects:

- **Performance Expectancy** captures the expected personal benefit regarding his job performance and is defined as "the degree to which an individual believes that using the technology will help him or her to attain gains in job performance."
- **Effort Expectancy** captures the expected effect on the effort in their jobs, which is defined as "the degree of ease associated with the use of the technology."
- **Social influence** captures social factors influencing the use of the technology and is defined as "the degree to which an individual perceives that important others believe he or she should use the technology."
- **Facilitating conditions** captures technical, administrative, or other factors influencing the use of the technology and is defined as "the degree to which an individual believes that an organizational and technical infrastructure exists to support use of the technology."
- **Behavioural intention** captures the intended usage of the technology in the future and is defined as "the degree to which a person has formulated conscious plans to perform or not perform some specified future behaviour."

Besides the core factors of UTAUT, we also analyzed associated factors such as attitude towards, anxiety, and self-efficacy with the technology:

- **Anxiety** captures negative emotions regarding the technology and is defined as "evoking anxious or emotional reactions when it comes to performing a behaviour (e.g., using the technology)."
- **Attitude** towards using technology captures positive or negative emotions regarding the technology and is defined as "an individual's overall affective reaction to using the technology."
- **Self-Efficacy** captures the impression made by the technology regarding ease of use and is defined as the "judgment of one's ability to use the technology to accomplish a particular job or task."

UTAUT states that the first four factors play a significant role as direct determinants of user acceptance and usage behaviour. As depicted in the next figure, the factors gender, age, experience, and voluntariness are confounding to this influence.

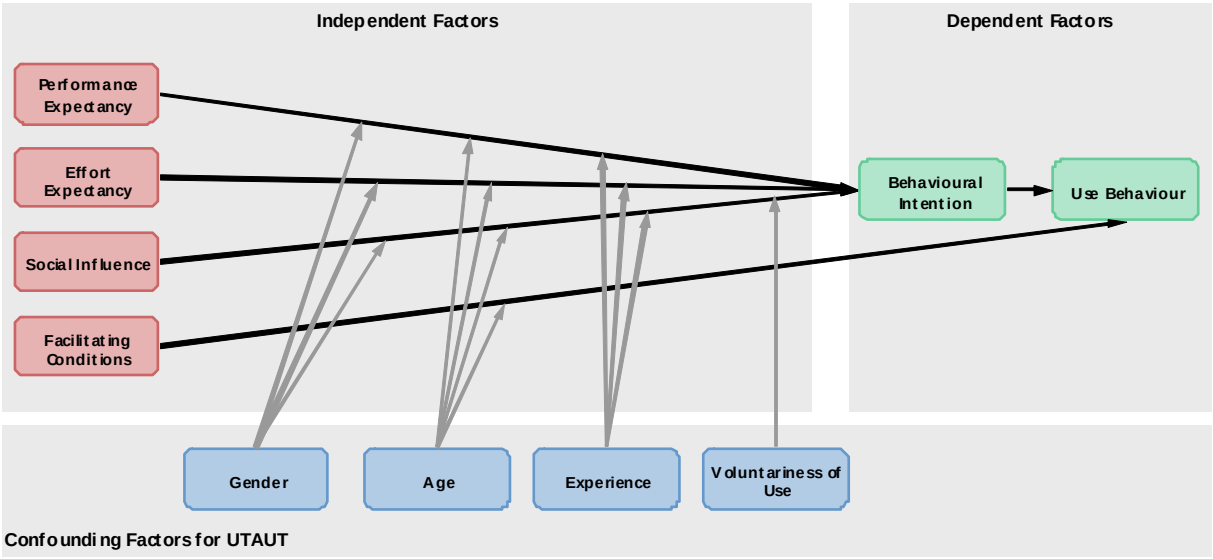

Figure 21: Model for the Unified Theory of Acceptance and Use of Technology (UTAUT)

In order to measure the subjects feedback regarding the items used in the questionnaire multiple choice questions were used wherever possible. They were based on a seven-level Likert scale later encoded from 1 (strongly disagree) to 7 (strongly agree), as these are more likely to be answered, and it is easy to statistically analyze the answers.

Furthermore, the UTAUT (Venkatesh et al., 2003) base instrument was adapted to the experiment by exchanging every mentioning of “system” by “VIDE-DD” in every item (i.e., question). Besides the core factors (PE, EE, FC, SI, and BI) used in UTAUT, we also collected information on related factors used in the UTAUT evaluation (AX, AT, and SE).

In order to evaluate if QDD technology such as VIDE-DD is accepted and will be used, we collect the following data:

- The questions (items) for the UTAUT factor BI, the predictors PE, EE, SI, and FC as well as the associated factors AX, SE, and AT.
- The moderators of UTAUT: gender (G), age (A), experience (E), and voluntariness (V)
- The UTAUT factors will be calculated using the associated items, if the load onto the factor. Furthermore, experience will be measured using several questions (see questionnaire 0).

The indirect factor used to evaluate the hypothesis H_{1d} will be the UTAUT factors and especially BI, while individual factors will also be evaluated.

3.4.1.4 Plan & Execution

As VIDE-DD cannot be compared to similar systems we will compare it against a manual inspection of a PIM regarding the implemented quality defects using two different models of software systems. These systems have to be improved by the subjects in two different runs of software developers. As depicted in the next figure, the first run uses System1 (SalesScenario) and VIDE-DD while the control group in the second group manually inspects System2 (BelAMI system). Both used the same base of 11 quality defect currently implemented in VIDE-DD.

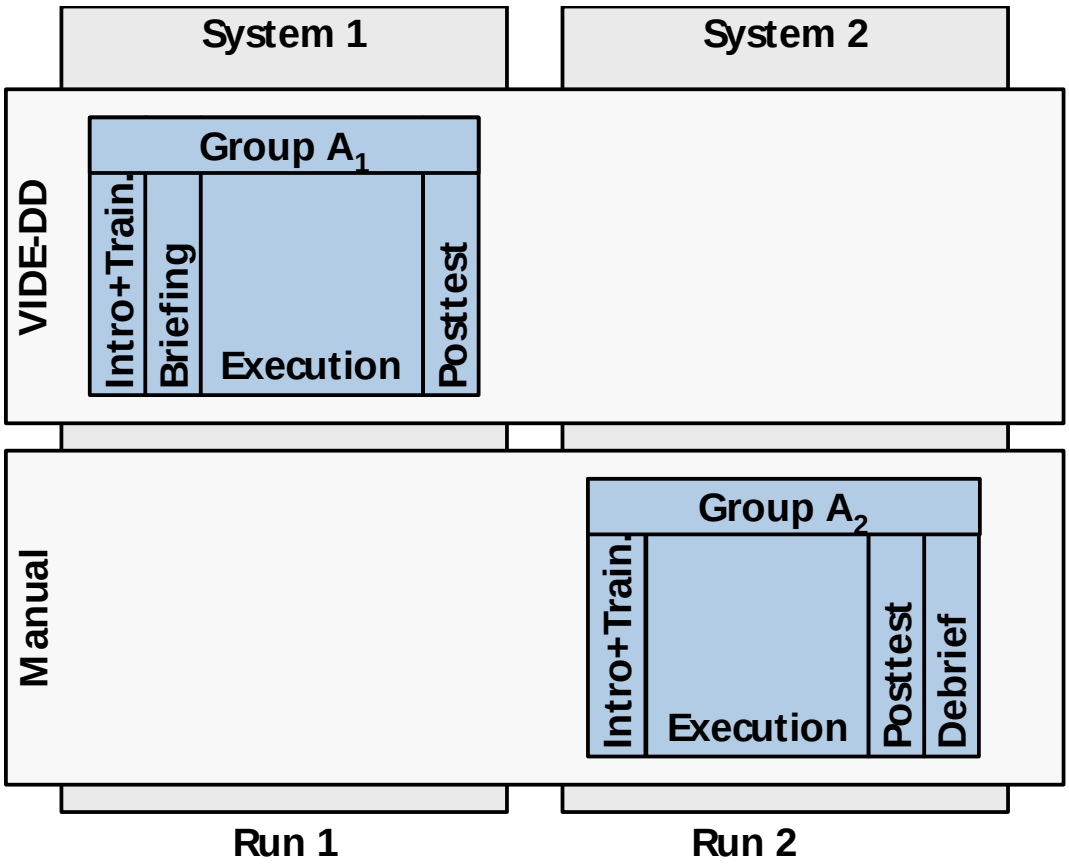

Figure 22: 1x2 factorial design of the experiment (incl. timeline)

Due to this design a **learning effect will BelAMI/EMERGE**, however, as VIDE-DD is used in the first run and the manual approach in the second run this will only improve the results of the manual approach.

3.4.1.5 Subjects

The experimental subjects used were all software developers and software engineering scientists, which studied computer science at the University of Kaiserslautern (3), University of Heidelberg (1), and the University of Applied Sciences in Mannheim (2) both in Germany.

3.4.1.6 Materials

3.4.1.6.1 Analyzed systems

- **System 1 (SalesScenario):** For the first evaluation the SalesScenario system was used, which was developed in the VIDE project as a demo system. It operationalizes the Opportunity Management step within a Customer Relationship Management (CRM) process.
- **System 2 (BelAMI System):** For the second evaluation the BelAMI system was used. The goal of this system is to support elderly people with innovative BelAMI/EMERGEncy monitoring and prevention approaches using ambient and unobtrusive sensors.

3.4.1.6.2 Diagnosed Defects

During the evaluation of VIDE-DD the following quality defects were used. The three thresholds represent three priority levels low, normal, and high.

- **Long Method**: A method with far too many statements (resp. actions) to be understandable and which probably tries to do too much on its own. The diagnosis was based on the number of semicolon with a threshold of 15.
- **Large Class**: A class with too many attributes and methods to be understandable and which probably tries to combine too many responsibilities. The diagnosis was based on the number of methods with a threshold of 9.
- **Lazy Class**: A class with too few attributes or methods and, therefore, doesn't pull its weight. The diagnosis was based on the number of methods with a threshold of 3.
- **Clingy Class**: A class with too many relations (e.g., associations or generalizations) with other classes within the system. The diagnosis was based on the number of relationships with a threshold of 6.
- **Long Parameter List**: A method with too many parameters to be understandable. The diagnosis was based on the number of parameter with a threshold of 4.
- **Comments**: While comments are great, refactoring may make most comments superfluous. If an element is heavily commented, it probably is too complex and may need refactoring. The diagnosis was based on the lines of comments with a threshold of 7.
- **Data Class**: Dumb data holders may provide better functionality if given more things to do. The diagnosis was based on the number of operations (with a threshold of 0), attributes (with a threshold of 1), getter-operations, and setter operations.
- **Indecent Exposure**: A public attribute that unnecessarily expose the internals of a class. The diagnosis was based on the number of public operations (with a threshold of 6), public attributes (with a threshold of 3), getter-operations, and setter operations.
- **Swiss Army Knife**: Excessively complex class that attempts to provide for too many possible uses of the class. The diagnosis was based on the number of implemented roles with a threshold of 6.
- **Type Embedded In Name**: A name where the type of an element within the system is embedded (and have to be changed). Often it is also redundant and forces you to change the name if the type changes. The diagnosis was based on the number of types (names) within the elements name with a threshold of 0.
- **Uncommunicative Name**: An abbreviated or extremly short name that might be misunderstood. A name should clearly describe what a method, class, or attribute is good for? The diagnosis was based on the length of the elements name with a threshold of 3 (using a whitelist for common term such as "i" or "id").

3.4.1.6.3 Info Materials

Previous to the actual experiment the subjects were briefed regarding software quality, quality defects, refactoring, and quality aspects. The material, which was available during the whole study, explained all used quality defects, methods of detection, and potential treatments.

3.4.1.6.4 Questionnaires

- Before the first run the subjects were briefed about the study and completed a **briefing questionnaire** about their background. This pretest was aimed at capturing information on the expertise of the subjects with the technologies used, their experience with programming, as well as their understanding of the task.

- After every runs a **posttest questionnaire** was conducted in order to collect information on the influence of the experiment on the subjects as well as their opinion about the perceived quality improvement (using ISO9126).
- After both runs a **debriefing questionnaire** had to be completed, which captured their opinion about the system (using the UTAUT instrument), the study, as well as other factors, such as motivation.

3.4.2 Data Preparation & Assessment

3.4.2.1 Data Quantification

The answers of the subjects on the paper-based briefing, posttest, and debriefing questionnaires were transferred to Excel (2003) and then SPSS (version 15) - using the following mapping:

- Yes/No Answers: yes = 1, no =0
- Agreement scales: Highest Agree = 7; Highest Disagree = 1
- Experience Scales: Highest Experience = 7; Highest Unexperience = 1
- Other Scales: Highest Agree = 7; Highest Disagree = 1

3.4.2.2 Anomaly Analysis

Not always are the values given by the subjects correct, unambiguous, and readable. They might even forget, overlook, or decide to ignore a question. However, probably due to the low amount of subject we did not have any missing or incomplete data and no ambiguous answers were found.

3.4.2.3 Data Set Reduction & Reliability Analysis

After preparing the data, they need to be analyzed if several items load onto the more general dimensions and which of them can be used to construct these factors in order to reduce the further analysis and testing efforts.

3.4.2.3.1 Reliability Analysis

The reliability of the identified components (resp. factors) about the subjects experience from the briefing questionnaire was checked using cronbach's alpha. The following table shows the *cronbach's alpha* values (based on standardized values) for the five factors regarding the subjects **experience** levels.

Table 21: Reliability analysis of factors for experience levels

Disturbing Factor	Factor Name	Cronbach's Alpha	Reliability	N of Items
exp_dev	Experience Development	0.77	High	4
exp_jp	Experience Java Programming	0.88	High	6
exp_ref	Experience Refactoring	0.95	High	4
exp_qa	Experience Software Quality Assurance	0.93	High	5
exp_main	Experience Software Maintenance	0.95	High	6

The results show that the PCA produced six highly reliable factors, which will be used in further statistical evaluations.

Similarly, the **UTAUT** instrument on the debriefing questionnaire were tested if they load on the "higher" factors. While UTAUT is a proven instrument to measure the acceptance and use of technology, it was not used in a context such as this controlled experiment. The table below shows the *cronbach's alpha* values (based on standardized values) for the eight factors of UTAUT regarding the subjects acceptance and potential use of a technology such as VIDE-DD.

Table 22: Reliability analysis of factors of UTAUT

Dependent Factor	Factor Name	Cronbach's Alpha	Reliability	N of Items
PE	Performance expectancy	0.55*	Low	4
EE	Effort expectancy	0.85	High	4
SI	Social influence	0.82	High	4
BI	Behavioural intention	0.98	High	3
FC	Facilitating conditions	0.45*	Low	4
AX	Anxiety	0.93	High	4
AT	Attitude toward VIDE-DD	0.82	High	4
SE	Self-efficacy	0.51*	Low	4

The results show that UTAUT is a fairly reliable instrument with three low, three medium, and one highly reliable factors. The Cronbach Alpha of the factors marked with an Asterisk (*) can be increased by removing respective items. Thus, the items PE1, FC4, and SE1 were removed in order to increase the Cronbach alphas at a level close to 0.7.

Finally, the **ISO9126** quality sub-characteristics on the post-test were tested if they load on the "higher" factors. While the characteristics and sub-characteristics of ISO9126 are not designed as an instrument for a questionnaire, the loading of sub-characteristics to their characteristics were checked as they are supposed to represent them. The table below shows the *cronbach's alpha* values (based on standardized values) for the six factors (resp. quality characteristics) of ISO9126 regarding the subjects perceived quality improvement of a technology such as VIDE-DD using the abovementioned quality defects.

Table 23: Reliability analysis of factors of ISO 9126

Dependent Factor	Factor Name	Cronbach's Alpha	Reliability	N of Items
QM_E	Maintainability	0.50*	Low	5
QP_E	Portability	0.65*	Medium	4
QE_E	Efficiency	N.A.		2
QR_E	Reliability	0.86	High	3
QF_E	Functionality	0.99	High	5
QU_E	Usability	0.95	High	3
QM_C	Maintainability	0.62*	Medium	5
QP_C	Portability	0.74	High	4
QE_C	Efficiency	N.A.		2
QR_C	Reliability	0.91	High	3
QF_C	Functionality	0.99	High	5
QU_C	Usability	0.58*	Low	3

The results show that using ISO9126 sub-characteristics in a questionnaire results in highly reliable instrument with one low and five highly reliable factors, which will be used in further statistical evaluations. For QE_E and QE_C (Efficiency) the Cronbach Alpha is not available

because there are only two items. The Cronbach Alpha of the factors marked with an Asterisk (*) can be increased by removing respective items

3.4.2.4 Outlier Analysis

Tukey's method was used to identify outlier based on the box-and-whisker plots (boxplots) to visually identify potential outliers. The analysis of outliers was limited to **experience**. We did not find an outlier, thus all data sets are used for the further evaluation.

3.4.2.5 Descriptive Statistics

Graphical, tabulatory, and summarative description of the data is needed for the identification of outliers, the distribution model, visual analytics, or to better understand the data. The following tables describe the measured data of all subjects as well as for each group. The means can be used for a first evaluation regarding differences of the dependent factors (in UTAUT and ISO9126). However, confounding factors such as experience are still not integrated.

Skewness is a measure of the lack of symmetry. A data set is symmetric if it looks the same to the left and right of the center point.

Kurtosis is a measure of similarity to a normal distribution. A data set with high kurtosis has a distinct peak near the mean, declines rather rapidly, and has low tails. A Data set with low kurtosis has a flat top near the mean rather than a sharp peak, declines slowly, and has high tails.

Data for the factor **experience** are described in the table below. As these are independent resp. confounding factors they stand for themselves and are not further analyzed.

Table 24: Descriptive statistics for experience factors

Factor	Factor Name	N	Min	Max	Mean	Std. Dev.
exp_dev	Experience Development	7	0	1	0.68	0.37
exp_jp	Experience Java Programming	7	2.33	5.67	4.4	1.20
exp_ref	Experience Refactoring	7	1	5.25	3.29	1.7
exp_qa	Experience Software Quality Assurance	7	1	4.8	2.66	1.37
exp_main	Experience Software Maintenance	7	1	4.33	2.62	1.51

The descriptive statistics for the **UTAUT factors**, as presented in the table below, show that the subjects don't perceive the system as a burden (effort expectancy). There is a positive attitude towards a system such as VIDE-DD (Attitude), and an expected performance boost (performance expectancy). The participants are neutral regarding the availability of an environment to use it (facilitating conditions) and regarding the future use of such a (behavioral intention).

Table 25: Descriptive statistics for UTAUT factors

Factor	Factor Name	N	Min	Max	Mean	Std. Dev.
PE	Performance expectancy	5	3.33	6	4.6	0.95
EE	Effort expectancy	5	4	5.8	5.14	0.76
AT	Attitude toward VIDE-DD	5	3.5	6	4.76	1.10
SI	Social influence	5	2.3	4.5	3.76	0.84

FC	Facilitating conditions	5	2.67	5.33	4	1.05
SE	Self-efficacy	5	2.67	4.67	3.73	0.80
AX	Anxiety	5	1	3.3	2.02	0.83
BI	Behavioural intention	5	1	6	4	1.87

The data for the **ISO9126 factors** in the table below show that the experimental group perceives the improvement of all factor except maintainability of the system better than the control group. Other quality aspects are almost identical with a slightly lower value than average – an expected expression as mostly maintainability-oriented quality defects were used and the subject are more likely to state that these other quality aspects are not improved.

Table 26: Descriptive statistics for ISO9126 factors

Factor	Factor Name	N	Min	Max	Mean	Std. Dev.
QM_E	Maintainability (Experiment)	7	2	6.33	4.23	1.44
QP_E	Portability (Experiment)	7	3	6.33	4.43	1.29
QE_E	Efficiency (Experiment)	7	1	6.5	3.93	2.20
QR_E	Reliability (Experiment)	7	1	6	3.97	1.64
QF_E	Functionality (Experiment)	7	1	6	3.6	1.91
QU_E	Usability (Experiment)	7	1	6.7	4.96	1.95
QM_C	Maintainability (Control)	5	3	6.25	4.9	1.26
QP_C	Portability (Control)	5	2	4.8	3.78	1.06
QE_C	Efficiency (Control)	5	1	4.5	2.9	1.75
QR_C	Reliability (Control)	5	1	5	3.4	1.52
QF_C	Functionality (Control)	5	1	4.8	3.64	1.51
QU_C	Usability (Control)	5	4	6	4.7	0.84

The data for the **refactoring protocol factors** for the first day (the "own" DCGA system) in the table below show that the experimental group investigated and treated more quality defects than the control group (Table 28).

Table 27: Descriptive statistics for Refactoring protocol factors (experiment group)

Factor	Factor Name	N	Min	Max	Mean	Std. Dev.
IC_E	# Classes investigated	7	8	24	16.72	6.04
IE_E	# Elements investigated	7	9	26	18.14	6.57
IQD_E	# Quality Defects investigated	7	10	64	31.71	18.80
Treats_E	# Treatments conducted	7	7	35	17.86	11.47
IgQD_E	# QDs ignored	5	1	23	13.6	9.34
TPQD_E	Treatments per QD	5	0.35	1	0.61	0.28
IPQD_E	Ignores per QD	5	0.04	0.58	0.37	0.20
QDPC_E	QDs per Class	7	1.2	2.7	1.8	0.56
TPC_E	Treatments per class	7	0.5	1.9	1.1	0.52
IPC_E	Ignores per class	5	0.1	1	0.66	0.36

While slightly more elements were investigated by the experimental group, the ratio of treatments (TPQD) vs. ignore activities (IPQD) are almost identical. Additionally, more quality defects per class were investigated and more treatments and ignore activities per class were conducted.

Table 28: Descriptive statistics for Refactoring protocol factors (control group)

Factor	Factor Name	N	Min	Max	Mean	Std. Dev.
IC_C	# Classes investigated	5	6	22	11.8	6.38
IE_C	# Elements investigated	5	8	38	15.8	12.56
IQD_C	# Quality Defects investigated	5	8	38	17.8	11.67
Treats_C	# Treatments conducted	5	2	27	10.2	10.59
IgQD_C	# QDs ignored	5	2	8	4.6	2.61
TPQD_C	Treatments per QD	5	0.14	0.88	0.53	0.35
IPQD_C	Ignores per QD	5	0.13	0.62	0.30	0.20
QDPC_C	QDs per Class	5	1.2	2	1.48	0.36
TPC_C	Treatments per class	5	0.2	1.2	0.76	0.47
IPC_C	Ignores per class	5	0.2	0.7	0.44	0.24

3.4.3 Subject Analysis

In order to evaluate the transferability to other environments and comparability of the results with other studies we analyzed the background and pre-existing knowledge of the subjects. Here we used the briefing questionnaires (0) in order to elicit information from them.

As the questionnaires asked several questions about the personal background, experience and knowledge of the subjects. In the following section the answers from these questions are further analyzed and evaluated.

3.4.3.1 Background data

The background data from the briefing questionnaire consists of six main information blocks. It includes information on the number of semesters, the number of practical courses already completed, the type of study conducted, as well as the major and minor subject of the study.

Six participants have a background in informatics, whereas on participant has a background in education. However, this participant has 20 years of experience in programming.

3.4.3.2 Experience of the Subjects

The briefing questionnaire was also used to identify the experience of the subjects with the main topics of the experiment. Based on the principal component analysis (PCA) of the questions on the questionnaire, five experience factors were identified and confirmed. These factors are experience with development, experience with Java programming, experience with refactoring, experience with software quality assurance, and experience with software maintenance. These factors were measured using a Likert scale. However, in raw years the following data was also measured:

Table 29: Descriptive statistics for experience of subjects

Factor	N	Min	Max	Mean	Std. Dev.
How many years of computer programming experience do you have, if any?	7	6	20	11.29	5.25

How many different applications have you programmed?	7	3	20	9.43	6.13
How many different applications have you programmed in Java?	7	0	15	7.86	5.93
How many years were you involved in maintaining & improving a software system?	7	0	3	0.86	1.07

As it can be seen in above table, the participants are quite experienced regarding software development. However, they are not very experienced regarding maintenance.

3.4.4 Hypothesis Testing

3.4.4.1 Impact on Quality – H_{1a}

The effect of the prioritization used in SQuaH on the perceived quality improvement was the main aspect of the controlled experiment. To investigate this effect of the independent factors on the dependent factors the hypotheses have to be statistically analyzed. The hypotheses $H_1^{Q\text{-}M}$,$H_1^{Q\text{-}E}$, $H_1^{Q\text{-}P}$, $H_1^{Q\text{-}R}$, $H_1^{Q\text{-}F}$, and $H_1^{Q\text{-}U}$ will be analyzed using a dependent sample t-test.

- H_{1a}: Using VIDE-DD has a *positive effect on the model quality* compared to the manual inspection of the software model (in the same timespan).

Table 30: Dependent Sample t-test for ISO9126 factors (experimental group vs. control group)

Dependent Factor	Factor Name	t	df	Sig. (2-tailed)	Std Error Mean	95% Confidence Interval		Power
						Lower	Upper	
QM	Maintainability	-1.04	4	0.36	0.99	1.72	-3.79	N.A.
QP	Portability	0.83	4	0.46	0.43	1.54	-0.83	N.A.
QE	Efficiency	0.50	4	0.64	1.00	3.28	-2.28	N.A.
QR	Reliability	0.32	4	0.77	0.50	1.55	-1.23	N.A.
QF	Functionality	-0.99	4	0.38	0.60	1.08	-2.28	N.A.
QU	Usability	0.21	4	0.84	1.15	3.42	-2.94	N.A.

Concluding from the results of the t-test, we are not able to reject the null-hypothesis.

3.4.4.2 Diagnostic Efficiency – H_{1b}

SQuaH as realized in VIDE-DD is aimed at improving the goal-oriented presentation of quality defects to software engineers. Beside an improvement of the software's quality it should also improve the treatment of the software. To investigate this effect on the dependent factors, hypotheses $H_1^{T\text{-}IC}$, $H_1^{T\text{-}IQD}$, and $H_1^{T\text{-}TQD}$ have to be statistically analyzed. In order to do that a dependent sample t-test is used.

- H_{1b}: Using VIDE-DD results in *more quality defects diagnosed* compared to the manual inspection of the software model (in the same timespan).

Table 31: Dependent Sample t-test for ISO9126 factors (experimental group vs. control group)

Dependent Factor	t	df	Sig. (2-tailed)	Std. Error Mean	95% Confidence Interval		Power
					Lower	Upper	
Number of Classes investigated	2.25	4	0.09	2.31	11.62	-1.22	0.34
Number of Quality Defects investigated	3.33	4	***0.03***	3.54	21.63	1.97	0.4
Number of Treatments conducted	0.95	4	0.40	6.13	22.81	-11.21	N.A.

The results from the t-tests show that the participants investigated significantly more quality defects when they were using the VIDE-DD. In addition, the number of investigated classes can also be considered as significant, assuming a one-tailed significance, which would result in a p-value of 0.04. Due to the low number of participants the power is quite low. A raise in the number of participants, e.g., to a total sample size of 12, would yield an acceptable power, higher than 0.8 in both cases.

However, the results do not show any significant difference regarding the number of treatments conducted.

3.4.4.3 Handling Efficiency – H_{1c}

- H_{1c}: Using VIDE-DD results in *more quality defects handled* compared to the manual inspection of the software model (in the same timespan).

Table 32: Dependent Sample t-test for handling efficiency (experimental group vs. control group)

Dependent Factor	t	df	Sig. (2-tailed)	Std Error Mean	95% Confidence Interval		Power
					Lower	Upper	
Handling Efficiency	1.62	3	0.20	0.06	0.30	-0.1	N.A.

Concluding from the results of the t-test, we are not able to reject the null-hypothesis. There is no significant difference regarding handling efficiency when using the VIDE-DD or doing the task manually.

3.4.4.4 Acceptance and Use of VIDE-DD (UTAUT) – H_{1d}

In order to investigate the effect of the independent factors on the dependent factors the hypotheses have to be statistically analyzed. The hypotheses $H_1^{A\text{-}BI}$, $H_1^{A\text{-}PE}$, $H_1^{A\text{-}EE}$, $H_1^{A\text{-}AT}$, $H_1^{A\text{-}SE}$, and $H_1^{A\text{-}AX}$ will be analyzed using a one sided t-test as the used measures are based upon an interval scale.

- H_{1d}: Quality defect diagnosis tools (such as VIDE-DD) have a significant *positive rating towards acceptance* and intention to be used.

After the independent and confounding factors are checked, the actual hypotheses are to be evaluated. The results of the normality analysis showed that SI and AT do not have a normal distribution and have to be tested using non-parametric tests.

Using a one-sample t-test against the value 4 we get the results as listed in the table below. They show that the UTAUT factors regarding effort expectancy and anxiety have a significant

difference from the neutral value 4. The positive factors performance expectancy has a positive difference from 4 and the negative factor Anxiety has a negative difference. In addition the power for these factors is high enough to accept the hypothesis.

For the other factors, the test did not yield significant differences.

Table 33: One Sample t-test for UTAUT factors against test value 4

Dependent Factor	Factor Name	t	df	Sig. (2-tailed)	Mean Differ.	95% Confidence Interval		Power
						Lower	Upper	
PE	Performance expectancy	1.41	4	0.23	0.60	-0.59	1.79	N.A.
EE	Effort expectancy	3.35	4	***0.03***	1.14	0.20	2.08	1.00
AT	Attitude	1.54	4	0.2	0.76	-0.61	2.13	N.A.
SI	Social influence	-0.64	4	0.56	-0.24	-1.29	0.81	N.A.
FC	Facilitating conditions	0.00	4	1.00	0.00	-1.31	1.31	N.A.
SE	Self-efficacy	-0.75	4	0.5	-0.27	-1.25	0.72	N.A.
AX	Anxiety	-5.36	4	***0.01***	-1.98	-3.01	-0.95	0.93
BI	Behavioural intention	0.00	4	1.00	0.00	-2.32	2.32	N.A.

3.4.5 Conclusions

Based on the results from the empirical study we can state the following statements:

- 80% more defects are found using VIDE-DD than manual inspections.
- VIDE-DD is perceived as useful, especially regarding its potential to reduce the effort required for the refactoring
- The use of QDD tools such as VIDE-DD are significantly influenced by the factors effort expectancy and anxiety.

We cannot state that VIDE-DD improves the quality significantly. However, the indications go in that direction. The low number of participants, and the fact that two participants did not finish the second part of the experiment for scheduling reasons, might have yielded these results. We assume that in practice the effects will be more visible.

In general, we can extract the following information from the UTAUT factor analysis:

- QDD Technology is perceived to have a low effort to use (EE) and is not intimidating for software engineers. However, the influence on the other factors is neither significantly positively nor negatively affected.
- The original UTAUT model does not perform very well in this evaluation. Explanations for the bad performance might be the low number of subjects, the supportive resp. assisting nature of QDD Technology, or cultural differences (UTAUT was evaluated in the context of companies in the United States).
- The best model to predict the intention to use QDD Technology is based upon Attitude (AT), Self-Efficacy (SE), and Social Influence (SI).

While the logical model significantly explaines 63.1% of the variance in BI, the results should be seen as preliminary. More subjects are needed for the evaluation of this model, however, as

no information of the basic population of software engineers exists, so far, the exact amount and mixture of subjects cannot be stated.

3.4.5.1 Future work

A future replication of the experiment using participants from industry would be helpful in order to provide more realistic data.

3.5 Evaluation of the aspect-oriented Modelling approach in VIDE (FIRST)

One part of the VIDE project was the investigation of the aspect-oriented techniques to allow for suitable modularisation of crosscutting concerns and to integrate the aspect weaving activity into the model driven development processes of the VIDE tool chain.

The VIDE environment allows for using aspect-oriented features (at PIM level and partially also at textual level) to realize so called crosscutting concerns in a modular way and to adapt the behaviour of a base application in a non-invasive way. The required aspect-oriented constructs are defined in an UML Profile. The aspect model, which makes use of the AO-related constructs, is stored in a separate model file. Hence, the base model has not to be modified or adapted.

The Aspect Weaver consumes the base model with the basic application and the aspect model with the additional behaviour and produces one merged object-oriented model at the same abstraction level, which covers the functionality of the base application as well as the additional functionality, originally specified within the aspect. The resulting model conforms to the VIDE metamodel and can be processed by further components without support of aspect-oriented semantics (e.g. execution engine, model compiler).

The approach was partially implemented in the context of the work package 9. The UML profile for representing aspect-oriented constructs was fully implemented. The more complex aspect weaver component was fully implemented for one scenario. This scenario includes the matching of different join point types (operation call action, read structural feature action, write structural feature action, etc.) and the weaving of VIDE conform aspects, using the binding kind *around*. This is the most complex case, so that all other cases can be implemented in an analogous way without additional problems. The aspect weaver component was developed using new model-based technologies, such EMF and ATL (http://www.eclipse.org/m2m/atl/). The corresponding eclipse plug-in was developed using Java, which realizes the seamless integration and the launching of the ATL model transformation engine.

To sum up, we have developed a extension to the VIDE tool chain, which provides the capabilities for modelling crosscutting concerns in a modular way using aspect-oriented concepts and which provides a fully VIDE compliant and seamless integrated aspect weaver.

Since the aspect weaver supports the one mentioned scenario, it is not possible to verify the measurable results in a competition between VIDE programmers and programmers using other development tools as stated in the description of work. The fact, that the aspect-oriented components do not implement all proposed concepts (new technology, low resources) would adulterate the results.

Therefore the measurable results will be verified in the following survey.

3.5.1 Measurable results

The measurable result, stated in the description of work, can be spitted into two separate parts, which of course partially relate on each other. In the following, the two parts are described.

MR3a: *Expressiveness in terms of join point models and possibilities of structural and behavioural adaptation compared to existing AOP approaches.*

How will it be verified?

The achieved AO modelling and composition facilities will be compared with existing AOP approaches at programming language level, considering in particular the adaptation semantics and expressiveness at PIM level.

Description:

One part, this criteria focus on, is the join point model, which describes the points/events in the execution of a system. These events can be intercepted and the behaviour can be adapted or modified. The types of join points with different properties define the join point model.

Since different AOP approaches can have different join point models, the expressiveness of these approaches with respect to the selectable join points can vary. Typical join point types are for example invocation of methods, execution of methods, exceptions. Join point types, which relate to the control statements (e.g. if-condition join point or loop join points in LogicAJ2) are usually not included in the join point model.

To capture certain join points, a pointcut selection mechanism is required. Pointcut selection mechanisms can differ in the flexibility and expressiveness. The most approaches tries to find a compromise between flexibility and understandability. Pointcut languages can quantify over static information (lexical structure, containment, static typing, inheritance, additional meta information, etc.) and over dynamic information available at runtime such as call stack, dynamic argument values, the full execution trace of the application, the structure of the dynamic heap, etc.

In the ALPHA approach (Ostermann et al., 2005), for instance, pointcuts are perceived as logical queries on different models or structures, such as AST, Execution trace and Heap.

The second part, the criteria focus on, is the possibility of structural and behavioural adaptation in a non-invasive way. For the behaviour adaptation, the following issues play a relevant role:

- Binding kind
- Mechanisms for managing advice precedence and aspect interference
- Static restriction of modules to be adapted
- Kinds of aspect instantiation
- Available context information
- …

The structural adaptation is about extending the structure of different modules in the program, such as adding new attributes and methods.

MR3b: *The (visual) aspect composition at PIM-level will be sufficient to cover at least 90% of typical AOP-scenarios in those parts of an application, that can be defined in the VIDE language.*

How will it be verified?

Representative scenarios will be chosen for the comparison with respect to the business application domain.

Description:

The requirements with respect to the features, an AO approach have to provide, depend on the scenarios. Each scenario may make use of different features.

The measurable result MR3a will be verified as follows. The key features, related to the comparison, of the WP3 approach will be briefly described. After that, different approaches for aspect-oriented programming will be introduced and compared to our aspect-oriented modelling approach.

To verify the measurable result MR3b, some typical scenarios will be defined. The features required for realization of the scenarios will be mentioned. A discussion will follow, weather our approach is suitable for the realisation of the proposed scenarios. Here, of course, we have to consider the fact, that a scenario can be implemented in different ways.

3.5.2 Comparison of aspect-oriented approaches

Various aspect-oriented approaches are available as extensions for different programming languages and frameworks. The AOP approaches, chosen to be described and compared in this section, represent a small subset of the existing approaches and differ in various characteristic features. The following description focuses on the key features and does not give a full introduction to the aspect-oriented languages.

3.5.2.1 AOM approach in VIDE

In work package 3, an approach for modelling and weaving aspects was developed. The developed concepts for aspect-orientation at PIM level were inspired by AspectJ (AspectJ 2008) and the AOM approach, presented by Fuentes and Sanchez (Fuentes et al.). We focussed on behaviour adaptation of base models in a non-invasive way. This kind of adaptation is more complex and consists of an explicit join point model, an approach for defining pointcuts and a weaving mechanism for merging aspect models and base models. Therefore a mechanism for structural adaptation was not developed, however our approach can be easily extended to support the structural modification of modules within the base model. For this reason, the example scenarios, such as the Consistency Check example, were designed in such a way, that the structural adaptation is not required.

Structural adaptation can, for instance, be used to store additional context information in an object, which is intercepted by an advice. Since we support different instantiation mechanisms (one aspect instance per system, once aspect instance per each callee or caller, one aspect instance per each join point), in some cases this requirement can be realized by storing the context information within the aspect instance.

Our pointcut model, in fact, could include any kind of action, but we have explicitly restricted it to the typical behavioural join points, like invocation of an operation, execution of an operation, setting and reading an attribute.

We have decided to allow for modelling of named and therefore independent advices, which can be explicitly bound to an advice. This feature allows for reusing advices in different contexts. An explicit binding (called also *Connector* in other approaches) is used to connect the advice and the pointcut. The binding defines the binding kind, such as *before*, *after* and

around. In general, the binding kinds *after* and *before* are special cased of the binding kind *around*, but the separation of the binding kinds simplifies the handling in several cases.

As already mentioned, the binding connects the advice, containing the additional behaviour to the pointcut. The pointcut defines the join points to be selected and consequently where the additional behaviour has to be injected in the base model. Similar to AspectJ, our pointcut mechanism makes use of different so called pointcut expressions. The pointcut expressions are represented as detailed metamodel instances, so that no textual parser is required. This decomposition of pointcuts facilitates the pointcut matching process using model-driven technologies.

Our AOM approach offer a various set of pointcut expressions and possibilities (see D3.2), which can be used to describe the join points, to be selected. To allow for passing context information into the advice, the advice itself as well as the pointcut contains a signature. The pointcut signature can be seen as a uniform interface to the selected join points and the corresponding contexts. The so called context exposure pointcut expressions are used to define, which context information should be passed to which argument in the advice signature. The available context information is the caller object, the callee object and the arguments of the captured join point.

Within the advice we offer the possibility to invoke explicitly the capture join point and pass the corresponding argument to it. Our approach does not offer the capability to get reflective information about the join point, because this feature was not required by the initially defined scenarios. In our approach we also do not handle the advice precedence and the aspect interference.

In our AOM approach, the corresponding aspect weaver provides a static weaving approach at the platform independent model level. The aspect and base models are at the same abstraction level as the woven model. This kind of weaving is analogous to the source code weaving at programming language level. The weaving approach consists of two phases: Pointcut matching and aspect weaving. The Pointcut matching process tries to find all points, which match to the pointcut declaration. If the pointcut definition makes use of properties, which are only available at runtime, the pointcut matching mechanism only can find potential join points in a static way. Another approach is an approximation of runtime properties using a static program analysis.

Since our approach do not support dynamic pointcut designators like *cflow*, all join points (respectively the corresponding model elements) are located during the pointcut matching. The aspect weaving process does not weave additional conditions into the base model, but rather focuses on the waving/injecting of the additional behaviour.

Our weaving mechanism does not inline the additional behaviour into the base model, but rather encapsulates the aspect in a separate class. Invocations of the advice methods are woven at the places, where the join points were identified. Since the aspect is encapsulated in a separate class, different instantiation mechanisms according to the modelled aspect properties can be realized in a more elegant way.

3.5.2.2 AspectJ

The most known and widespread AOP approach is AspectJ (AspectJ 2008), which is realized as an aspect-oriented language extension to Java. AspectJ provides the functionality for behavioural and structural adaption of the base system. Structural adaptation is realized using Inter-type declarations. Using this feature, the programmer can declare attributes and methods

which cut across multiple classes and interfaces. Furthermore, Inter-type declarations can be used to modify inheritance relationship between classes.

For reasons, which were already mentioned, our approach does not support structural adaptation. The realisation of structural adaptation for platform independent models would result in a straightforward model merging mechanism, so that this feature could be integrated easily.

AspectJ provides a dynamic join point model. Join points are certain well-defined points in the execution of the program and not program elements at source code level. Each invocation of a method, for example, is a separate join point, even if this point is represented by only one program element in the source code. The corresponding program elements at source code level are called join point shadows. AspectJ provides not only for kinds of join points like method call, method execution, field get and field set (AO approach in VIDE), but also includes join point kinds like exception handling and advice execution (For complete reference see AspectJ documentation (AspectJ 2008)).

The pointcut language of AspectJ provides pointcut designators to specify static and dynamic properties and to compose and reuse pointcuts. Similar effect is achieved by our composition of pointcut expressions using the *IntersectionPCE*. In contrast to our approach, advices are bound directly to pointcuts and cannot be reused in a suitable way. The *proceed* statement, which allows the invocation of the captured join point can be invoked without arguments, if the arguments were not passed to the advice. This is very useful for advices without any modifications to the arguments. This convenient argument buffering mechanism is not realized in the VIDE AO approach.

Since the AspectJ approach allows for using dynamic pointcut designators, the join points cannot be fully resolved in a static way. The AspectJ compiler checks the static properties during the pointcut resolving. The result is a set of *potential* join points. During the aspect weaving phase, corresponding behaviour for checking the runtime properties is woven before the advice call. Only if the woven condition is true at runtime, the advice is executed (Vanderperren et al., 2005). This mechanism for weaving additional conditions or explicit behaviour for identifying join points at runtime is necessary, if the pointcut mechanism shall support the specification of dynamic pointcut properties.

AspectJ is one of the first commonly used AOP approaches. A multitude of aspect-oriented languages are inspired by the functionality of AspectJ. Hence, the descriptions of the following aspect-oriented languages will focus on the significant differences.

3.5.2.3 JBoss AOP

JBoss AOP (JBoss 2008) is an aspect-oriented extension to Java. It was developed in the context of JBoss, but still can be used without the whole JBoss application server context. Overall, JBoss AOP offers similar features like AspectJ. JBoss AOP is not only a framework, but also a prepackaged set of aspects that are applied via annotations, pointcut expressions, or dynamically at runtime. Some of these include caching, asynchronous communication, transactions, security, remoting, etc. Similar to our approach, JBoss AOP use named advices and named pointcuts, so that these advices can be bound/connect in a flexible way. The aspects and interceptors (aspect with only one advice) are defined as pure Java classes. To get context information within an advice, a special API is used (ConstructorInvocation, MethodInvocation, FieldWriteInvocation, etc.). The pointcuts and the bindings are defined either using a XML descriptor or as Java annotation.

JBoss AOP provides a lot of useful features (explicit precedence handling, dynamic pointcut designators, the ability to add and remove advice bindings at runtime, Introductions, etc.), but one of the most important advantages with respect to our approach is the capability to quantify over additional information, added as Java annotation to all constructs, which can serve as a join point. Using the Java annotations, the programmer can add additional semantics to the program elements. This additional meta information can be used in combination with the properties of the program element to specify pointcuts. This approach partially makes the pointcuts robust against some program evolution.

3.5.2.4 JAsCo

JAsCo (Vanderperren et al., 2005; JAsCo 2008) in general is an advanced AOP approach, hence, only the key features will be discussed (see references for detailed information).

The JAsCo approach focuses mainly on reusability of aspects amongst multiple applications. Crosscutting concerns are modularized in so called *Aspect Beans*, which do not refer to concrete component types and APIs. The aspect beans are fully independent of the base code. On the other side, a *Connector* instantiates an aspect bean within a concrete component context. The Connector approach is similar to our binding, but works on aspect level and not on advice level. The Connector serves as a deployment mechanism. The Pointcut-Advice combinations are defined in so called *Hooks* which are a special kind of inner classes of aspect beans. Pointcuts are defined within the hook constructor in an abstract way. They do not refer to concrete program elements, but rather to a kind of template, which is defined in the hook constructor signature. The concrete pointcuts are defined within the connectors, which instantiate the hooks and pass the concrete pointcuts to them.

To be more concrete, the pointcut definition obviously depends on the base code (lexical structure, containment, etc.). To achieve the reusability of such aspect beans, JAsCo excludes the part, which deals with the lexical information, from the aspect bean. This fact makes the aspect bean independent of the concrete lexical information and therefore more abstract. The structural part of the pointcut is defined within the hook (part of the aspect bean), the lexical part of the pointcut is defined within the Connector and is than passed to a concrete instance of the corresponding hook.

The join point model is similar to AspectJ. JAsCo provides a lot of features such as customisation of hooks, introductions (called Virtual Mixins), precedence mechanism and combination strategies, which are able to manage the cooperation of aspects (see JAsCo language description (JAsCo 2008)).

As a solution for the problem of the *Jumping Aspects* (Brichau et al., 2000), JAsCo provides a very powerful mechanism, which allows for matching complete sequences of events rather than singular events. This approach is called stateful aspects. These approach bases on the state machine construct. Abstract Pointcuts serves as events, which trigger the transitions between specified states. Advices can be bound to states. Summing up, the occurrences of join points, which are captured by a pointcut, trigger the transition of the specified state machine, but do not directly cause an invocation of an advice. The advice is invoked not before a bound state is reached or entered. The listing 1 gives an impression, how to define stateful pointcuts (Example taken from http://ssel.vub.ac.be/jasco/documentation:stateful)

```
package test;

class ExampleLoggerBean {
    hook StatefulHook {

        StatefulHook(startmethod(..args1), runningmethod(..args2),stopmethod(..args3)) {
            start>p1;
            p1: execution(startmethod) > p3||p2;
            p3: execution(stopmethod) > p1;
            p2: execution(runningmethod) > p3||p2;
        }

        before p2 (){
             //do the logging:
             System.out.println("executing: "
                 +calledmethod.getName()+" "
                 +calledmethod.getClassName());
        }
    }
```

Figure 23: Example of a Stateful Pointcut

Our approach does not support such stateful aspects, but to some extent, it is possible to "simulate" this behaviour by implementing the state machine logic within an aspect manually. This is shown for example by the M4JPDD generator for AspectJ. Using JPDDs (Stein et al., 2006), stateful and pattern-based pointcuts can be modelled. Some of the modelled constructs do not have the corresponding counterpart in AspectJ, so that the addition logic is generated as a state machine implementation within the aspect.

3.5.2.5 LogicAJ

This approach also addresses the problem, that aspect-oriented languages in general increase the modularity but negatively influence evolvability by introducing lexical dependencies to the base program (Kniesel et al., 2005). In comparison to out approach and to AspectJ, LogicAJ introduces a concept called *Logic Meta-Variables* (LogicAJ 2008). A meta-variable captures one base program element. Meta-variables are like named wildcards, but in contrast to the wildcards for instance in AspectJ or in our approach, one can refer to the value matched by a meta-variable by using the same meta-variable in another expression within the same scope. All occurrences of the same meta-variable in a scope must match the same value. This mechanism is used also by the JPDD pointcut modelling approach. In this approach, it is possible to refer to values by using the same meta-variable (identifier) in further expressions. JPDD therefore allows for expressing stateful pointcuts, which makes use of value-based dependencies between different events.

The example in the listing below gives an impression, how the logic meta-variables are used.

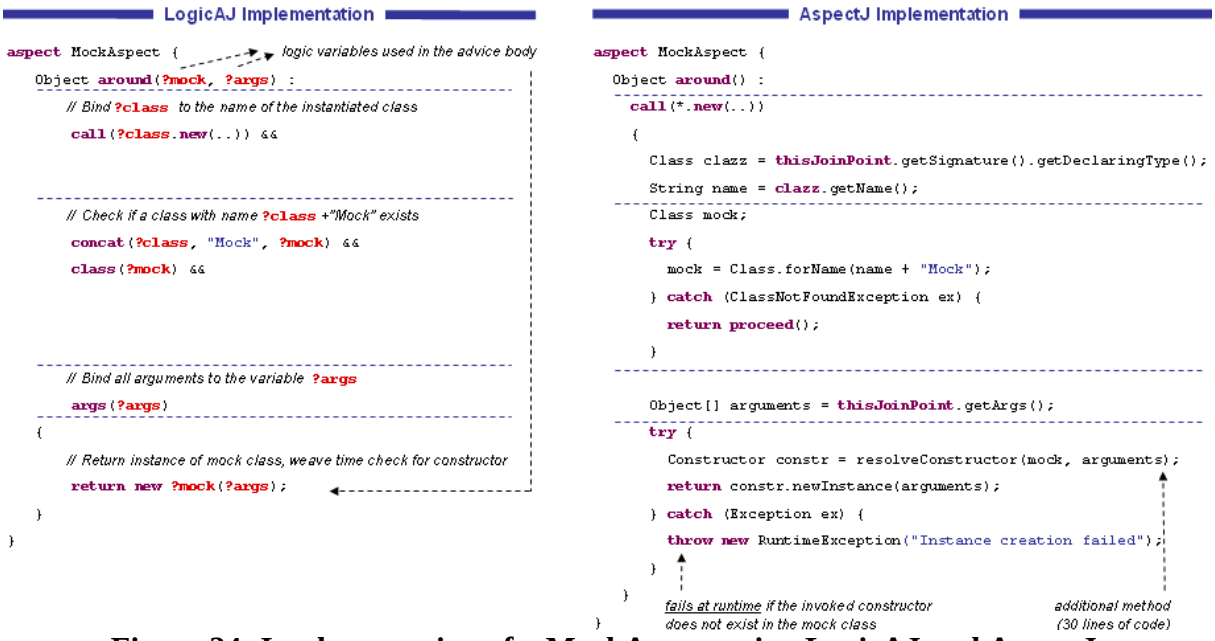

Figure 24: Implementation of a MockAspect using LogicAJ and AspectJ

Furthermore, LogicAJ provides an advanced mechanism for the analysis of interferences between aspects (LogicAJ 2008).

3.5.3 Typical AO Scenarios

The following sections will give an overview of the scenarios to be used for the evaluation of our approach. Features, required for the corresponding realisation will be mentioned and it will be discussed, whether the scenario can be realised in sufficient way. This survey focuses on the following categories:

- Join point model
- Expressiveness of the pointcut language
- Structural adaptation
- Behavioural adaptation
  - Context information
  - Modification of the context passed parameter
  - Invocation of the captured join point
- Binding

Criteria like reusability and aspect interference are hard to measure and go beyond the scope of this survey.

3.5.3.1 Monitoring

Monitoring can be done on various levels and for different purposes. Commonly used middleware platforms already provide built-in capabilities for monitoring. On application level, often an individual monitoring mechanism has to be developed. What and when should be monitored, depends on the individual scenario. Mixed code/model which contains the business logic as well as the monitoring functionality is difficult to maintain, increases the

complexity and decreases the understandability. Monitoring is a typical crosscutting concern so that aspect-oriented concepts can be used for a concrete realisation.

A possible process model for Performance Monitoring can be described as follows:

1. Define performance targets
2. Derive/select metrics for performance measurement
3. Identify required measurement points
4. Inject Code/behavioural model for measuring
5. Define monitoring scenarios
6. Integrating of monitoring

From the aspect-oriented point of view, during step 2 and 3, the aspect oriented approach plays a significant role. After the definition of the measurement points, the aspect-oriented approach has to be checked, if the join point model provides the required types of points in the execution of the system. If the join point model, for instance, only consists of join points for field access, it is not possible to capture method invocations.

Furthermore, the capabilities of the pointcut mechanism, which allows for selecting join points using different information, decides, whether the defined measurement points can be captured. For instance, if we want to select single method invocations to be monitored, we only need a simple pointcut mechanism. But for monitoring of complete execution patterns, an advanced pointcut mechanism (see e.g. JAsCo) is required.

If the identified measurement points do not have similar properties on model or code level, the pointcut mechanism cannot select the required points by quantifying over model or code properties (e.g. lexical structure, containment, inheritance, cflow, etc.). In this case, additional meta information for the measurement points in the model is required. This information has to be inserted manually into the model at the corresponding places. The pointcut mechanism than have to provide the capability for selecting join points based of these meta information.

On the other hand, the monitoring can be realized in such way, that all join points are calling a main aspect, which implements the logic concerning for instance the type of the join point, the name of the method and the caller/callee of the method. This alternative requires the AO approach to provide extensive context information of join points within the advice.

Regarding step 4, our weaving mechanism is suitable for behavioural adaptation of the model. The additional behaviour is injected into the base model according to the waving strategy, described in deliverable D3.2.

Considering all the mentioned issues, a common monitoring functionality can be realized using the VIDE AOM approach.

3.5.3.2 Logging

Logging aspects log several activities and events of the system and can for instance be used for Tracing, Debugging and Profiling. The applicability of the selected AO approach depends strongly on the issues to be logged and on the join point model and the pointcut mechanism.

On the one hand, the aspect (what and when should be logged) can be on the one hand realized using an advanced pointcut mechanism without any logic within the aspect. On the

other hand, the simple pointcut mechanism can capture all possible join points and the aspect logic decides by using context information of the join point, whether to log the activity.

A logging mechanism with standard requirements can be implemented using the VIDE AO approach.

3.5.3.3 Error handling

Similar to logging, the error handling aspect observes/logs the activities and events of the system. However, the error handling aspect focuses on the observation of errors, occurred in the system. Often, error handling aspects base on the join points, triggered by exceptions. Possible behaviour can be realized within an error handling aspect, such as:

- Sett attributes to uncritical values
- Switch to safety state
- Dump the system state and shut down the system
- …

The behaviour, executing during the error handling can be expressed and woven into the base model using the VIDE approach, but our join point model does not support exception join points, so that the error handling based on exception join points cannot be realized.

3.5.3.4 Security

Security is a typical application area for aspect-oriented concepts. Security can be realized on different levels and can contain for instance authentication and authorization. The kind of security, which is described in this section, focuses on the access authorisation for various methods.

If we want to ensure, that only users with the correct permissions are allowed to invoke a method, the aspect has to check the current user and his permissions. If the user is not logged in or do not have the right permissions, the method is not executed.

To achieve the effect, that a method is executed in a conditional way, the aspect-oriented approach needs to offer the binding kind *around*, which allows replacing the method invocation with an advice invocation. Furthermore, in case of the right permissions, the invocation of the intercepted method has to be allowed. The aspect has to check the intercepted method, the user and his role. Using this information, the aspect decides whether to execute the intercepted method or rather to raise a corresponding exception. This look-up can be done completely within the aspect using the context information of the intercepted method and a kind of look-up table for allowed users/roles for this method. In this case, the AO approach has to provide the capability to determine the name of the intercepted method. On the other hand, the information with respect to the roles having the permissions for executing the methods can be added directly into the code or model as meta information. In this case, the aspect has only to check, whether the current user is associated to the required role. Otherwise, a corresponding exception is raised.

Summing up, the AO approach in VIDE do not provide support for evaluation of additional meta information (neither for pointcut matching nor within the advice). Nevertheless our approach can be used for realizing the security aspect without meta information in the base model. The whole logic can be implemented within the corresponding advice.

3.5.3.5 Business rules / Consistency Checks

In the domain of business applications, certain business rules may not be violated during the lifecycle of the business application. A special case of such business rules is our example dealing with consistency checks of several business objects.

Several consistency checks are performed when the state of the opportunity business object or some of the associated objects changes. The actions and activities that enforce the consistency checks cut across different modules. The following listing depicts an example of consistency check (for detailed description, see D3.1 and 3.2).

```
(C3): Opportunity.processStatusValidSinceDate <
        SalesForecast.expectedProcessingDatePeriod.StartDate
```

The consistency checks are performed when a relevant value or state of a business object is going to be modified. From the point of view of the AO approach, these points in the execution of the system have to be captured. If the join points are captured and the advice performing the consistency check is executed, it is necessary to get access to the involved business objects to perform the consistency checks. For this purpose, the capability for context exposure has to be provided.

Various consistency checks exist for various business object types. It has to be decided, which consistency check have to be applied for which business object. On the one hand, individual pointcuts can contain the type checking, so that the bound advice only consists of the corresponding consistency check for one business object type. On the other hand, a general pointcut can capture all points relevant to the consistency check concern. All rules for the consistency checks are described in one advice, which also contains the logic for resolving the type of the business object and the corresponding consistency check to be applied.

Both cases can be realized using our aspect-oriented modelling approach.

3.5.3.6 Measurement of code coverage using AOP

Coverage measurement is a relevant mechanism, to get an idea, which program or model elements are covered by a test suite. This mechanism can be used during test execution or for the test generation process to achieve the selected coverage criteria. Various coverage criteria on source code and on model level are used. Often, this functionality is realized using code instrumentation.

Again, if we, for instance, want to measure statement coverage for method invocations and/or field accesses using aspect-oriented concepts, our join point model, described in D3.2, seems to be sufficient. On the other side, criteria like condition coverage cannot be analysed using our approach, because we cannot intercept the evaluating of conditions e.g. within “if” statements.

The LogicAJ2 (LogicAJ2 2008) approach provides a very generic code pattern base pointcut mechanism. The listing in figure below (taken from http://roots.iai.uni-bonn.de/research/logicaj/logicaj2/pointcuts) shows the possibility to match an “if” statement with the corresponding condition.

```
pointcut if_with_bool_condition( ?jp , ?condition , ??stmts ):

    binary_boolean_expressions( ?condition , ?_ , ?_ ) &&
    stmt ( ?jp , if ( ?condition ){ ??stmts} );
```

Figure 25: Pointcut for "if" statements

3.5.4 Conclusion

The comparison of the aspect-oriented approaches has shown, that the approach, developed in the VIDE project provides features, which are similar to the features provided by the AspectJ approach. The pointcut mechanism of our approach does not support dynamic properties like cflow and advanced mechanisms like stateful pointcuts. In contrast to AspectJ, the introduction of named advice and an explicit binding between pointcut and advice improve the reusability of aspects. The main objective of the workpackage 3 was not the implementation of a large variety of AO features but rather the integration of aspect-oriented concepts into model-driven software engineering processes and into the VIDE language, which makes use of a seamless integration of imperative and declarative language constructs. The flexibility of our approach provides the capability for various extensions such as modelling and matching of dynamic properties.

The second part of the survey introduces some of the typical AO scenarios and discussed the feasibility with respect to the VIDE AO approach. Some of the required features are missing (e.g. exception join point), but nearly all scenarios can be designed in such a manner, that the VIDE AO approach provides capabilities for a suitable realisation.

3.5.5 Future work

To make the AOM solution fully useful, from our point of view the following areas has to be addressed:

Aspect Editor:

The creation of aspects consists of three parts: Aspect structure, Advice behaviour and Pointcut definition. Since the structural part of the aspect is realized as UML stereotypes of existing elements, the aspect structure can be created using common UML tools, like Topcased. The advice behaviour is created using the textual editor, which was extended to advice-specific constructs. The pointcuts and the corresponding Pointcuts expressions are realized as additional meta classes and therefore not creatable by using the UML editor without a corresponding extension. In a further work (e.g. separate project) the following items could be addressed regarding the Aspect Editor:

- Creation of the advice behaviour using the visual editor
- Convenient creation of pointcut definitions
    - textual pointcut language and integration into the Textual Editor
    - visual pointcut language and integration into the Visual Editor
- Integration of existing approaches for modelling pointcuts (e.g. JPDD)
- Extension of the expressiveness of our Pointcut approach with respect to the specific features of the VIDE language (e.g. history and pattern based pointcuts)
- Using dynamic pointcut expressions, which refers to the dynamic/runtime properties of the behavioural model

- Additional tool support, such as
    - Refactoring of Aspect-Oriented behavioural models incl. update/rephrase of pointcut definitions
    - Automatic generation of pointcut definition from selected elements

Aspect Weaver:

In our prototype the aspect weaver supports only a subset of the proposed features. The following functionality could be addressed in further work:

- Completion of the proposed features
- Weaving of dynamic conditions into the behavioural model to allow support for dynamic properties with the pointcut language
- Realizing of different weaving strategies and a comparison of the resulting model with respect to different criteria (e.g. performance)
- Traceability between the aspect-oriented model (created and handled by the user) and the object-oriented result model (intended to be processed only by further components). This functionality is required if for example a generated object-oriented model produces errors during the execution and the errors have to be mapped to the aspect-oriented model, which was created by the modeller/user.

During the work on the aspect-oriented concepts in the VIDE context, some new research questions come into existence, which could be part of following projects:

- Behaviour adaptation in declarative constructs (OCL expressions), which do not represent the control flow in an explicit way
- The VIDE language provides initialization blocks, which can be used to initialize the attributes. In this declaration block only a subset of actions is allowed (e.g. AddStructuralFeatureAction). During the aspect weaving, additional behaviour cannot be injected in such initialization block, because we use invocations of advice methods to adapt the behaviour. The weaving mechanism has to be refined to consider these restrictions
- Investigate the integration of aspect-oriented constructs at CIM level

4 Vertical Evaluation

4.1 Introduction

The VIDE toolset is designed to assist in the process of the development of data intense business applications using a model driven approach. The components are outlined in a process view in Figure 26.

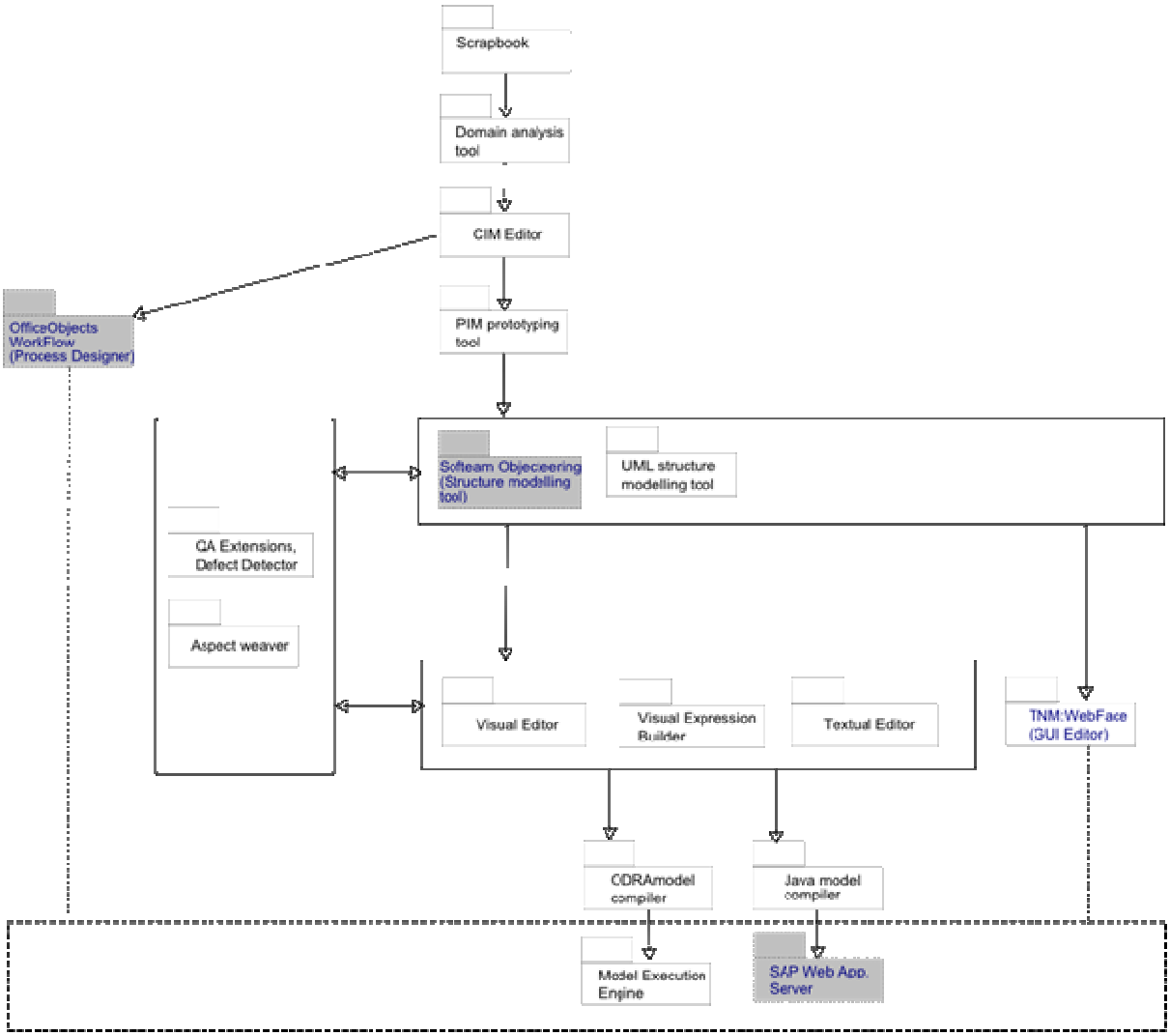

Figure 26: VIDE Components in a process view

The development of the toolset has been the result of research and prototype development by the VIDE consortium. The effectiveness of the toolset may be evaluated against the commercial offerings already in the market place and this report outlines this process. Section 4.2 outlines the value added features of VIDE for each of the MDA levels and Section 4.3 outlines ten of the major toolchains that have a similar scope to the VIDE toolset. Section 4.4 presents the tool chains feature by feature and discusses their coverage. The results are presented as a comparison matrix in Section 4.5 before concluding in Section 4.6.

4.2 The value added features of VIDE

4.2.1 Introduction

The VIDE toolset consists of a set of interconnected components which provide the stakeholders a number of unique features for data intensive systems development. This section outlines the features available at the different levels and identifies the stakeholders which are likely to find that feature-set useful. It should be pointed out that in some cases an individual component gives little added value, but seen in conjunction with further components the value-added is quite clear.

4.2.2 Pre-CIM level

4.2.2.1 Overview

The VIDE toolset provides among other things, a pre-CIM set of components which provide various functions. For example, the Scrapbook component allows users to store descriptions of the problem domain, including business requirements and operational procedures. The Analysis Palette allows the informal communication between domain specialists and business analysts.

4.2.2.2 Pre –CIM Stakeholders

The stakeholders at this level of the toolset are likely to be domain specialists, business managers and business analysts, using this part of the toolset as an aid to communication and to allow informal modelling.

4.2.2.3 The selection of features for evaluation

The Scrapbook component allows users to select parts of the domain description in order to build a visual (informal) scrap model as an initial attempt to comprehend the described problem domain issues.

An additional pre-CIM component is the Analysis Palette which allows for selection of scrapbook model elements for further analysis and construction of an Analysis model. One of the key tasks of analysis within the Analysis Palette component is the refinement of originally unclear (bloops) domain items (scraps) scrapbook elements into roles, activities, or data objects. The Analysis Palette component allows elements of the analysis model to be selected and exported for creating an initial business process model at the CIM level.

4.2.2.4 Evaluation criteria

Table 34: pre-CIM evaluation criteria

Pre-CIM	• Creating and storing descriptions of the domain (including descriptions of business requirements and operational procedures) • Selection of desired scrap text from the domain description to create a visual informal scrap model. • Selection of scrap model elements for further analysis and creation of an informal analysis model. • Refinement of domain model elements into roles, activities or data objects.

	• Selection of analysis model elements for use in creating an initial business process model

4.2.3 CIM Level

4.2.3.1 Overview

The CIM level component can either work stand-alone to create a VCLL model from scratch; or import the informal models created by pre-CIM tools and automatically create a first cut business process model. The model can be refined to include organizational structure, data structures and business rules. The process model is then either exported to create a workflow or a high level first cut UML2 model.

4.2.3.2 CIM Stakeholders

The stakeholders at this level of the toolset are the business analysts and maybe the systems analysts with knowledge of BPMN.

4.2.3.3 The selection of features for evaluation

The VCLL editor can import elements from the Analysis Palette and transform them into a basic VCLL model. A business analyst can now create a finer structure together with the domain expert in order to have a model understandable by non-IT-people but a good starting point for a PIM model. The next transformation is then made by the PIM prototyping tool. The VCLL consist of four views structuring the domain description, namely the process view, data view, organizational view and business rules.

4.2.3.4 Evaluation criteria

Table 35: CIM evaluation criteria

CIM	• Import of un-structured requirements information
	• Creation of Business Rules from both data structure and organisational structure model elements in addition to natural language.
	• Enhanced business process modelling using extended BPMN
	  ◦ Creating such things as multi-media artefact information
	  ◦ Creating data elements, organisational data and business rules inside the process. (These can be used singly or combined.)
	• Creation of organisational and data structures using

	elements helpful to the non-IT person. • Export of information from the process model as XPDL (for a workflow engine.) • Use of the export of a process model as a blueprint to allow creation of UML2 diagrams

4.2.4 CIM to PIM transition

4.2.4.1 Overview

The CIM to PIM transition is performed by using heuristics to transform the process model from the CIM editor into a high level UML2 model. This allows the analyst to refine the information to create more detailed models.

4.2.4.2 CIM to PIM Stakeholders

The stakeholders at this level of the toolset are business and systems analysts and analyst programmers.

4.2.4.3 The selection of features for evaluation

One of the main activities that follows business process modelling is the creation of a design model for a software system that would address parts of the business process. In VIDE, the PIM Prototyping (PPT) component provides functionality for designers to apply a set of heuristics to derive high level class models from a business process model automatically. Similarly, the PPT component provides functionality for the automatic derivation of a high level behaviour model (activity chart) from a given business process model.

4.2.4.4 Evaluation criteria

Table 36: CIM to PIM transition evaluation criteria

CIM to PIM transition	• Automatic creation of class model from business process model (by use of heuristics for BPM-to-Class mapping). • Automatic creation of activity chart from business process model (by use of heuristics for BPM-to-Activity mapping).

4.2.5 PIM level

4.2.5.1 Overview

PIM is the most extensively supported layer of VIDE tooling. It is aimed at achieving at the PIM level the degree of precision analogous to programming languages, but at the same time – to benefit from its being a model-level artifact. This means on one hand, having the

structure and behaviour stored in relatively abstract and flexible form – to make it easily explorable, traceable and transformable and possible to visualize, and, on the other hand, precise enough to test run it and to derive working code from it.

Hence, the core of VIDE PIM-level functionality is model repository compliant with a UML 2 tool for high-level modelling, and accompanied by UML compliant textual and visual language for precise behaviour specification. To take a bigger advantage of model executability, web service interfaces and wrappers to existing (XML and relational) data sources are provided. The flexibility of having detailed behaviour represented into UML model is demonstrated by PIM level aspect composition and quality defect detection tools, as well as platform-specific code generators.

4.2.5.2 PIM Stakeholders

The stakeholders at this level of the toolset are software modellers – analysts, PIM-level programmers and testers.

4.2.5.3 The selection of features for evaluation

4.2.5.3.1 Model execution at the PIM level

Having precise behavioural constructs implemented inside modelling tool and backed with executable semantics, allows not only to consider fully automated construction of target platform applications, but also test-running models inside the modelling tool using sample data. This way, an additional advantage of complete behaviour specification is exploited – testing a model and validating its behaviour with prospective users becomes much more straightforward: comparing to tools that do not involve detailed PIM-level behaviour specification – this avoids coding in the target platform's language. In turn, comparing to tools that support precise models but do not allow to directly running them – this allows to avoid the deployment at the target platform for testing.

4.2.5.3.2 Textual language and editor for UML action modelling

A modelling tool supporting executable models requires appropriate modelling constructs both in terms of underlying metamodel, as well as in concrete syntax. The most common and proven approach to providing this is defining a textual language with its mapping to the metamodel. As far as UML-based solutions are considered, this would involve a language that maps onto UML Actions and Activities. The lack of such language would force the user to manually construct low-level behavioural constructs in the UML metamodel instance which is definitely unproductive due to the effort involved.

4.2.5.3.3 Visual language and editor for UML action modelling

This feature refers to the availability of an appropriate visual notation for behaviour modelling with UML actions. Such an appropriate notation must provide a different visual representation of the different actions. Such a notation would simplify modelling with UML actions as the users do not have to work directly with the meta-model. The existence of such a notation without supporting tool is not helpful. For this reason, this feature refers to the availability of a visual editor that supports the visual notation.

4.2.5.3.4 OCL used as a high-level expression and query language

This feature directly refers to OMG standardized expression language. Putting it in more general terms, we are going to look for modelling languages means of semantically precise

specification of behaviour, offering a level of abstraction similar to the one of database query languages. Without this, the modelling language would constitute a step backward in terms of the level of abstraction compared to its platform-specific counterparts, with respective losses in terms of expressiveness and optimization possibilities.

4.2.5.3.5 OCL query expressions visualized

For dealing with non-trivial data structures, the expressiveness of expression language may be of high importance. Te context of a modelling tool, where significant amount of models, including behaviour is visualized, brings an additional incentive to consider query visualization. This would naturally complement visual representation of actions; it may be also used for ad-hoc queries construction. Visualization of queries allows to replace textual query language and provide more tool assistance when constructing an expression. Finally, a bit higher level of abstraction / declarativeness compared to the language like OCL can be achieved.

4.2.5.3.6 User-friendly action modelling

This feature refers to the availability of concrete syntaxes that shield the users from complexities of the UML Actions Meta-model. In fact, the meta-model defines several constructs such as object flows, control flows, input and output pin, etc. This makes programming with UML Actions very complex. The availability of a concrete syntax (visual or textual) simplifies this task by hiding the complexities of the abstract syntax.

4.2.5.3.7 PIM-level aspect-oriented modelling

The aspect-oriented extension provides the capability for modelling crosscutting behaviour in a modular way using aspect-oriented concepts. For this purpose, the UML-based VIDE metamodel was extended using the UML Profile mechanism. This profile allows for modelling required aspect-oriented constructs such as *aspects*, *pointcuts*, *advice*. The pointcuts are defined using so called pointcut expressions, provided also by the AO profile. The pointcut expressions provide a flexible approach for describing the model properties. All model elements, which conform to these properties, are called join points. The metamodel for the pointcut expressions is extensible so that required but not included features can be added easily.

The structural elements (aspect, advice, etc.) can be created using a UML conform modelling tool, such as *Topcased*. The advice behaviour can be specified using the Textual Editor, which was extended to provide advice-specific constructs.

Since the common tool chains in the MDA process do not support aspect-oriented concepts, this extension provides also an aspect weaver, which consumes the aspect-oriented models and produces object-oriented models, which conform to the VIDE metamodel. The input and the output models are at the same abstraction level (PIM level).

The modeller/developer at PIM level can benefit from the aspect-oriented modularization concepts (reduced complexity, increased maintainability, etc.), while the further components, like model execution engine and code generator, can process the woven models without providing AO support.

4.2.5.3.8 PIM-level quality defect detection

The VIDE Quality Defect Detector (VIDE-DD) is responsible for the automatic diagnosis and presentation of quality defects within platform independent models. It uses information about the software model, stored within the VIDE PIM repository, to analyze the model, diagnose

quality defects, and annotate the model using UML-Annotations. The information stored within these Annotations is then used by the quality defect presenting mechanism (an extension to the diagrams presentation mechanisms) to visually enrich the diagrams (within the VIDE visual editor) with information on these defects. These defects are symbolized using icons at the defective elements on the diagram, such as a class, a method, a relation or the diagram itself. Furthermore, VIDE-DD supports software modelers with the presentation of quality defects (prioritized according to a given quality model) within a list view.

The features of VIDE-DD sensiblize the software modellers about quality defects, focus the attention to important quality defects, improve the model quality if quality defects are removed, and result in a positive reception by software modellers.

The defect detection feature, as realized in the VIDE-DD system, consists of the following sub-features:

- Model Analysis
  - Support for single-characteristic defects
  - Support for multi-characteristic defects
  - Use of model metrics
  - Use of model characteristics (e.g., element identifier/names)
- Defect Diagnosis
  - Support for single-location defects
  - Support for multi-location defects (interconnection of defects)
  - Support to persist Ignore decisions
  - Support to (De-)Emphasize defect priority
  - Support to remove individual defects
  - Support to annotate Context Factors (Design Patterns, Data Class, …)
- Defect Annotation/Storage:
  - Annotation of all UML elements (Entities, Relations, …)
  - Use of EAnnotations within a diagram
- Defect Presentation:
  - Use of eclipse marker, eclipse decorators, and icons within a diagram
  - Defect differentiation by Type using different Icons
  - Defect Differentiation by Priority using color
  - Support for prioritization of defects (based on QM)
  - Support for nested defects (e.g., at Comments or Parameter) PIM-level publication of web services - modelling and execution

This feature refers to the ability to specify in the model that a class should be published as a web service and the ability to select the methods that will be exposed. This feature also includes the ability to generate all code artifacts that are necessary for deploying and running the published web service.

4.2.5.3.9 PIM-level consumption of web services - modelling and execution

This feature refers to the ability to model operation calls that target web services and not only classes of the local model. The feature includes the modelling of service consumption (i.e., the ability to specify a call operation action that interact with an external web service) and the execution of that service consumption (either directly or after code generation)

4.2.5.3.10 Platform independent means for GUI development

Graphical user interface implemented with WebFace can be published as Java applications or as Java Applets in a web browser. As a runtime environment, they only need a Java Runtime Environment, which is available for many operating systems (for example Windows, Linux, Solaris, Mac OS X). The same GUI can be used as an Applet or as an application without change, the GUI must only be published again.

4.2.5.3.11 Availability of a logical structure of GUI

In contrast to other available Java Development Environments (for example Netbeans), a GUI developed with TNM:WebFace provides an interface to the internal logical connections between the various graphical components. Traditional Java development environments implement the connections between the components with generated Java code. In a GUI generated with WebFace, these connections are implemented as a logical data structure, no code is generated in order to connect the components. This functionality can be used to implement complex help systems. During runtime, the help system can easily monitor the actions of the user by analyzing the events between the components. For example, in the research project Smartkom (www.smartkom.org), the graphical user interface was developed on base of TNM:WebFace. In this project, the data provided by the GUI was used to implement a graphical presentation agent, who guides the users through the running application.

4.2.5.4 Evaluation criteria

Table 37: PIM level evaluation criteria

PIM level	• Model execution at the PIM level • Textual language and editor for UML action modelling • Visual language and editor for UML action modelling • User-friendly action modelling • OCL used as a high-level expression and query language • OCL query expressions visualized • PIM-level aspect-oriented modelling • PIM-level quality defect detection • PIM-level publication of web services - modelling and execution • PIM-level consumption of web services - modelling and execution • Platform independent means for GUI development • Availability of a logical structure of GUI

4.2.6 PIM to PSM

4.2.6.1 Overview

The translation from PIM to PSM is the process of translating VIDE/UML static model with the whole behaviour of operations described in the VIDE PIM level language into a PSM model, taking into account language (platform) specific constructs like Java or ODRA.

4.2.6.2 PIM to PSM Stakeholders

The stakeholders at this level of the toolset are likely to be VIDE programmers.

4.2.6.3 The selection of features for evaluation

The evaluation will be based on the code generation feature of each tool. The structural code generation, i.e. generation of class skeletons, will not be evaluated but the focus will be on code generation of the behaviour. This behaviour can be described using high level (PIM) language like OCL or UML dynamic diagrams (Activity and Sequence).

4.2.6.4 Evaluation criteria

Table 38: PIM to PSM evaluation criteria

PIM-PSM	• Code generation of behaviour from PIM (method/constructor bodies) • Complete and compile-ready Java code generation

4.2.7 Conclusion

This section has highlighted the stakeholders and value added features of the VIDE toolset at each of the MDA levels from pre-CIM to the PIM transition level. Section 3 will investigate the toolchains which VIDE can be compared with, using these features.

4.3 Tool chains

4.3.1 Introduction

There are a large number of MDA tools available in the marketplace today. The Object Management Group has 58 committed companies listed on its web site, and the work in Deliverable 5 outlined the various features of these tools. There are fewer companies which offer tools that cover the whole spectrum from pre-CIM to code. To perform a fair tool comparison, it is necessary to select those tools that cover most of the development cycle and thus allow a direct comparison with VIDE. Thus, ten tool chains have been selected which are closest to the depth of functionality of VIDE. Some of these tool chains are well known and the list has been prioritised in order of their (to the consortium) perceived popularity amongst developers.

Table 39:Tool chain summary

Tool Chain	**Constituting Tools**			
	Pre-CIM	**CIM**	**PIM**	**Code generation**
1 - IBM	Rational Requisite Pro	Websphere Business Modeler	Rational Software Architect	Rational Software Architect
2 - Borland	Borland Caliber DefineIT	Borland Together	Borland Together	Borland Together
3 - Visual Paradigm	Visual Paradigm for UML	Business Process Visual Architect	Visual Paradigm for UML	Visual Paradigm for UML
4 - Telelogic	Telelogic DOORS	Telelogic System Architect	Telelogic Modeler/Rhapsody/Tau	Rhapsody/Tau
5 - Eclipse	Topcased	Eclipse STP BPMN	Topcased /STP SOA	Acceleo
6 - Artisan	Telelogic DOORS	Artisan Studio	Artisan Studio	Artisan Studio
7 - openAmeos	openAmeos or Telelogic DOORS	openAmeos	openAmeos	openAmeos
8 - NoMagic & InteractiveObjects	Magic Draw	Magic Draw	Magic Draw	Arcstyler

9 - Select Business Solutions	Select Solution for MDA	Select Architect	Select Solution for MDA	Select synchronizers
10 - MID	Innovator Business + Innovator Object	Innovator Business + Innovator Object	Innovator Object	Innovator Object

4.3.2 IBM

Vendor link: www.ibm.com/

Table 40: IBM tool chain availability

Tool	Version	License
	Link	
	Description	
IBM Rational RequisitePro	7.0.1	15 day trial available, registration required
	www-01.ibm.com/software/awdtools/reqpro/	
	-	
IBM Websphere Business Modeler	6.2	30 day trial available, registration required
	www-01.ibm.com/software/integration/wbimodeler/index.html	
	-	
IBM Rational Software Modeler	7.5	30 day trial available, registration required
	www-01.ibm.com/software/awdtools/modeler/swmodeler/index.html	
	-	
IBM Rational Software Architect	7.5	30 day trial available, registration required
	www-01.ibm.com/software/awdtools/swarchitect/websphere/	
	-	

Table 41: IBM Toolchain and MDA levels

MDA step	Tool
Pre-CIM, domain analysis	IBM Rational RequisitePro[1]
CIM: Business Process Modelling	IBM Websphere Business Modeler[3]

PIM-Modelling	IBM Rational Software Modeler (part of RSA)
Code generation	IBM Rational Software Architect[4]

1) Requirements Management: Drag-and-drop association of *RequisitePro*® requirements. It Create and leverage a domain specific modeling languages (DSMLs) to represent the unique business problem and solution domain.

3) Integration with *IBM WebSphere Business Modeler (WBM)*: The SOA for WebSphere transformations generate service component specifications and behaviors expressed as SDCL and BPEL, suitable for deployment to *WebSphere Process Server* and for further development and testing using *WebSphere Integration Developer* to create composite service applications.

4) Includes IBM® Rational® Application Developer and uses transformations to generate Java, JEE 5 and EJB code to be deployed on the IBM Rational Deployment

4.3.3 Borland

Vendor link: www.borland.com/

Table 42: Borland tool chain availability

Tool	**Version**	**License**
	Link	
	Description	
Borland Together	2008	**15 day trial available, registration required**
	www.borland.com/us/products/together/index.html	
	-	
Borland Caliber DefineIT	2008	**30 day trial available, registration required**
	www.borland.com/us/products/caliber/defineit.html	
	-	

Table 43: Borland Toolchain and MDA levels

MDA step	**Tool**
Pre-CIM, domain analysis	Borland Caliber DefineIT, Domain-Specific Language (DSL) Toolkit
CIM: Business Process Modelling	Borland Together[1]
PIM-Modelling	Borland Together
Code generation	Borland Together[2]

1) Business Process Modeling Notation (BPMN) with validation checking and

Import/export of BPEL for Web Services (BPEL4WS).

2) Code generators for Java, J2EE™, C++, and C#.

4.3.4 Visual Paradigm

Vendor link: www.visual-paradigm.com/

Table 44: Visual Paradigm tool chain availability

Tool	**Version**	**License**
	Link	
	Description	
Visual Paradigm Suite	3.4	http://en.wikipedia.org/wiki/Software_license**Proprietary with Free Community Edition (Registration required)**
	www.visual-paradigm.com/product/vpuml/	
	The *Visual Paradigm Suite 3.4* combines Visual Paradigm's development tools to visually and diagrammatically design, integrate and deploy enterprise applications. *Visual Paradigm Suite* can be integrated into several IDEs using the *Smart Development Environment* (SDE) adapter.	
Visual Paradigm for UML	6.4	http://en.wikipedia.org/wiki/Software_license**Proprietary with Free Community Edition (Registration required)**
	www.visual-paradigm.com/product/vpuml/	
	Part of *Visual Paradigm Suite*.	
Business Process Visual Architect	2.4	**Proprietary with Free Community Edition (Registration required)**
	www.visual-paradigm.com/product/bpva/	
	A business modelling tool designed for visualizing, understanding, analyzing, improving and documenting business processes, document flow and information in your organization (Part of *Visual Paradigm Suite*).	

The *Visual Paradigm Suite 3.4* combines Visual Paradigm's development tools to visually and diagrammatically design, integrate and deploy enterprise applications. *Visual Paradigm Suite* can be integrated into several IDEs using the *Smart Development Environment* (SDE) adapter.

Table 45: Visual Paradigm

MDA step	**Tool**

Pre-CIM, domain analysis	Visual Paradigm for UML[1]
CIM: Business Process Modeling	Business Process Visual Architect[2]
PIM-Modeling	Visual Paradigm for UML
Code Generation	Visual Paradigm for UML

1) Requirements analysis using Textual Analysis, CRC Card diagrams and Requirement diagrams.

2) A business modelling tool designed for visualizing, understanding, analyzing, improving and documenting business processes, document flow and information in your organization.

4.3.5 Telelogic

Vendor link: www.telelogic.com/

Table 46: Telelogic tool chain availability

Tool	**Version**	**License**
	Link	
	Description	
Telelogic DOORS	9.0	**No trial available.**
	www.telelogic.com/products/doors/index.cfm	
	-	
Telelogic System Architect	11.2	**15 day trial available, registration required**
	www.telelogic.com/products/systemarchitect/index.cfm	
	-	
Telelogic Modeler	4.2	http://en.wikipedia.org/wiki/Software_license**Free Download, registration required**
	www.telelogic.com/products/modeler/index.cfm	
	-	
Telelogic Rhapsody	7.3	**30 day trial available, registration required**
	www.telelogic.com/products/rhapsody/index.cfm	
	-	
Telelogic Tau	3.1	**30 day trial available, registration required**
	www.telelogic.com/products/tau/index.cfm	
	-	

Table 47: Telelogic Toolchain and MDA levels

MDA step	Tool
Pre-CIM, domain analysis	Telelogic DOORS
CIM: Business Process Modelling	Telelogic System Architect
PIM-Modelling	Telelogic Modeler / Telelogic Rhapsody / Telelogic Tau
Code Generation	Telelogic Rhapsody / Telelogic Tau

4.3.6 Eclipse

Vendor link: www.eclipse.org/

(www.eclipse.org/stp/ & www.topcased.org/ & www.acceleo.org/)

Table 48: Eclipse tool chain availability

Tool	Version	License
	Link	
	Description	
Eclipse SOA Tools Platform (STP) Project	1.0.1	http://en.wikipedia.org/wiki/Software_licenseOpen source
	www.eclipse.org/stp/	
	-	
Topcased	2.2.0	http://en.wikipedia.org/wiki/Software_licenseOpen source
	gforge.enseeiht.fr/frs/?group_id=52	
	-	
Acceleo	2.4.0	Open source http://en.wikipedia.org/wiki/Software_license
	www.acceleo.org/pages/download-acceleo/en	
	-	

Table 49: Eclipse Toolchain and MDA levels

MDA step	Tool
Pre-CIM, domain analysis	Topcased
CIM: Business Process Modelling	Eclipse STP Project
PIM-Modelling	Topcased
Code Generation	Acceleo

4.3.7 Artisan

Vendor link: www.artisansoftwaretools.com/

Table 50: Artisan tool chain availability

Tool	Version	License
	Link	
	Description	
Artisan Studio	7.0	**30 day trial available, registration required**
	www.artisansoftwaretools.com/products/artisan-studio/	
	Artisan Studio is an all-in-one integrated development tool suite which provides systems and software modeling and component based development.	
Automatic Code Synchronizer	-	**30 day trial available, registration required**
	www.artisansoftwaretools.com/products/tool-set/automatic-code-synchronizer/	
	Studio's *Automatic Code Synchronizer* (ACS) supports the continuous synchronization between model and code.	
Template Development Kit	-	**30 day trial available, registration required**
	www.artisansoftwaretools.com/products/tool-set/automatic-code-synchronizer/	
	Studio's *Template Development Kit* (TDK) enables the ability to extend the existing code generation capabilities offered by Artisan Studio, in order to make them fit any specific development process.	

Artisan Studio is an all-in-one integrated development tool suite which provides systems and software modelling and component based development.

Table 51: Artisan Toolchain and MDA levels

MDA step	Tool
Requirements analysis	Telelogic DOORS (Artisan's synchronizer)[1]
Pre-CIM, domain analysis	Telelogic DOORS (Artisan's synchronizer)[1]
CIM: Business Process Modelling	Artisan Studio
PIM-Modelling	Artisan Studio
Code generation	Automatic Code Synchronizer[2], Template Development Kit[3] (as part of Artisan Studio)

1) Artisan Studio includes a bi-directional *Telelogic DOORS* synchronization. This allows the creation modification or removal of requirements and traceability in either Studio or DOORS through the synchronization of the contents of the two tools.

2) Studio's *Automatic Code Synchronizer* (ACS) supports the continuous synchronization between model and code.

3) Studio's *Template Development Kit* (TDK) enables the ability to extend the existing code generation capabilities offered by Artisan Studio, in order to make them fit any specific development process.

4.3.8 openAmeos

Vendor link: www.openameos.org/

Table 52: OpenAmeos tool chain availability

Tool	**Version**	**License**
	Link	
	Description	
OpenAmeos	10.0	**Open source**
	www.openameos.org/download	
	-	

Table 53: OpenAmeos Toolchain and MDA levels

MDA step	**Tool**
Requirements analysis	OpenAmeos (or Telelogic DOORS[1])
Pre-CIM, domain analysis	Ameos/UML Profile support (or Telelogic DOORS)
CIM: Business Process Modelling	OpenAmeos
PIM-Modelling	OpenAmeos
Code Generation	Script Manager (as part of OpenAmeos)

1) The integration of Telelogic DOORS allows OpenAmeos objects to be sent to DOORS and allows DOORS to manage the details of linking these objects to requirements.

4.3.9 NoMagic & InteractiveObjects

Vendor link: www.nomagic.com/dispatcher.php & www.interactive-objects.com/

Table 54:NOMagic tool chain availability

Tool	**Version**	**License**
	Link	
	Description	
MagicDraw	15.5	http://en.wikipedia.org/wiki/Software_license**Community Edition available, Registration required**
	www.magicdraw.com/	
	-	
Arcstyler	5.5	**30 day trial available, registration required**
	www.interactive-objects.com/products/arcstyler/arcstyler-overview.html	
	-	

Table 55: MagicDraw Toolchain and MDA levels

MDA step	**Tool**
Pre-CIM, domain analysis	MagicDraw
CIM: Business Process Modelling	MagicDraw
PIM-Modelling	MagicDraw
Code Generation	Arcstyler

4.3.10 Select Business Solutions

Vendor link: www.selectbs.com/

Table 56: Select tool chain availability

Tool	**Version**	**License**
	Link	
	Description	
Select Solution Factory	-	**trial with reduced functionality available, registration required**
	www.selectbs.com/products/select-solution-factory.htm	
	Select Solution Factory is an integrated set of products and modeling tools for Component Based Development, design review, service/component management, requirements management, code generation and reuse.	
Select Solution for MDA	-	**trial with reduced functionality available, registration required**
	www.selectbs.com/products/select-solution-for-mda.htm	
	Upgrade of *Select Solution Factory.*	
Select Architect	-	**trial with reduced functionality available, registration required**
	www.selectbs.com/products/select-architect.htm	
	Part of *Select Solution Factory.*	
Select synchronizers	-	**trial with reduced functionality available, registration required**
	www.selectbs.com/products/select-jsync.htm	
	Part of *Select Solution Factory.*	

Table 57: Select Toolchain and MDA levels

MDA step	**Tool**
Pre-CIM, domain analysis	Select Solution for MDA
CIM: Business Process Modelling	Select Architect
PIM-Modelling	Select Solution for MDA
Code Generation	Select synchronizers (e.g. Select JSync)

4.3.11 MID

Vendor link: www.mid.de/MID-Homepage.startseite.0.html?&L=&L=1

Table 58: MID tool chain availability

Tool	Version	License
	Link	
	Description	
MID Innovator Object Edition	2008	**74 day trial available, registration required**
	www.mid.de/Object.inno_object.0.html?&L=1	
	As part of the modeling platform Innovator, Innovator Object is the flexible tool for object-oriented modeling with UML 2.	
MID Innovator Business Edition	2008	**74 day trial available, registration required**
	www.mid.de/Business.inno_business.0.html?&L=1	
	-	

As part of the modelling platform Innovator, Innovator Object is the flexible tool for object-oriented modelling with UML 2.

Table 59: MID Toolchain and MDA levels

MDA step	Tool
Pre-CIM, domain analysis	MID Innovator Object/ Innovator Business
CIM: Business Process Modelling	MID Innovator Object/ Innovator Business
PIM-Modelling	MID Innovator Object
Code Generation	MID Innovator Object

4.3.12 Conclusion

This section has presented the ten tools with which the consortium will compare VIDE. It has outlined how the different tool sets are combined to form a chain and the availability of the tools including their licensing arrangements. The next section takes each of the features outlined in Section 4.2 and compares them to the individual tool chains presented in this Section.

4.4 Vertical evaluation

This section takes each of the unique features of the VIDE toolset defined in Section 4.2 and compares them to each of the toolchain components at that level as defined in Section 4.3. Therefore each level shows an overview of the toolset and then describes the features and if/where they are found in the tools.

4.4.1 Pre-CIM

Table 60: Tool chain summary CIM Level

Tool Chain	**Constituting Tools**			
	Pre-CIM	**CIM**	**PIM**	**Code generation**
1 - IBM	**Rational Requisite Pro**	Websphere Business Modeler	Rational Software Architect	Rational Software Architect
2 - Borland	**Borland Caliber DefineIT**	Borland Together	Borland Together	Borland Together
3 - Visual Paradigm	**Visual Paradigm for UML**	Business Process Visual Architect	Visual Paradigm for UML	Visual Paradigm for UML
4 - Telelogic	**Telelogic DOORS**	Telelogic System Architect	Telelogic Modeler/Rhapsody/Tau	Rhapsody/Tau
5 - Eclipse	**Topcased**	Eclipse STP BPMN	Topcased /STP SOA	Acceleo
6 - Artisan	**Telelogic DOORS**	Artisan Studio	Artisan Studio	Artisan Studio
7 – openAmeos	**openAmeos or Telelogic DOORS**	openAmeos	openAmeos	openAmeos
8 - NoMagic & InteractiveObjects	**Magic Draw**	Magic Draw	Magic Draw	Arcstyler
9 - Select Business Solutions	**Select Solution for MDA**	Select Architect	Select Solution for MDA	Select synchronizers
10 - MID	**Innovator Business + Innovator Object**	Innovator Business + Innovator Object	Innovator Object	Innovator Object

4.4.1.1 Creating and storing descriptions of the domain (including descriptions of business requirements and operational procedures)

This function addresses the need to create descriptions of the problem domain, including operational procedures that often become an important reference point for analysts and developers during a software project.

4.4.1.1.1 Rational Requisite Pro

This component of the Rational suite addresses the need to record any type of textual information, including domain information. Requisite Pro enables users to build statements of requirements and assign priorities to such statements based on organisational needs. Requisite Pro, however, does not provide a means to directly use the information in any subsequent stage of the software project.

4.4.1.1.2 Borland Caliber DefineIT (BCD)

BCD is a requirements recording tool with a database backend that allows a tree structure for storing requirements, and their characterisation. Users can associate stakeholders to specific requirements. Targeted at business analysts, the tool provides a visual means for depicting the requirements as scenarios see Figure 27.

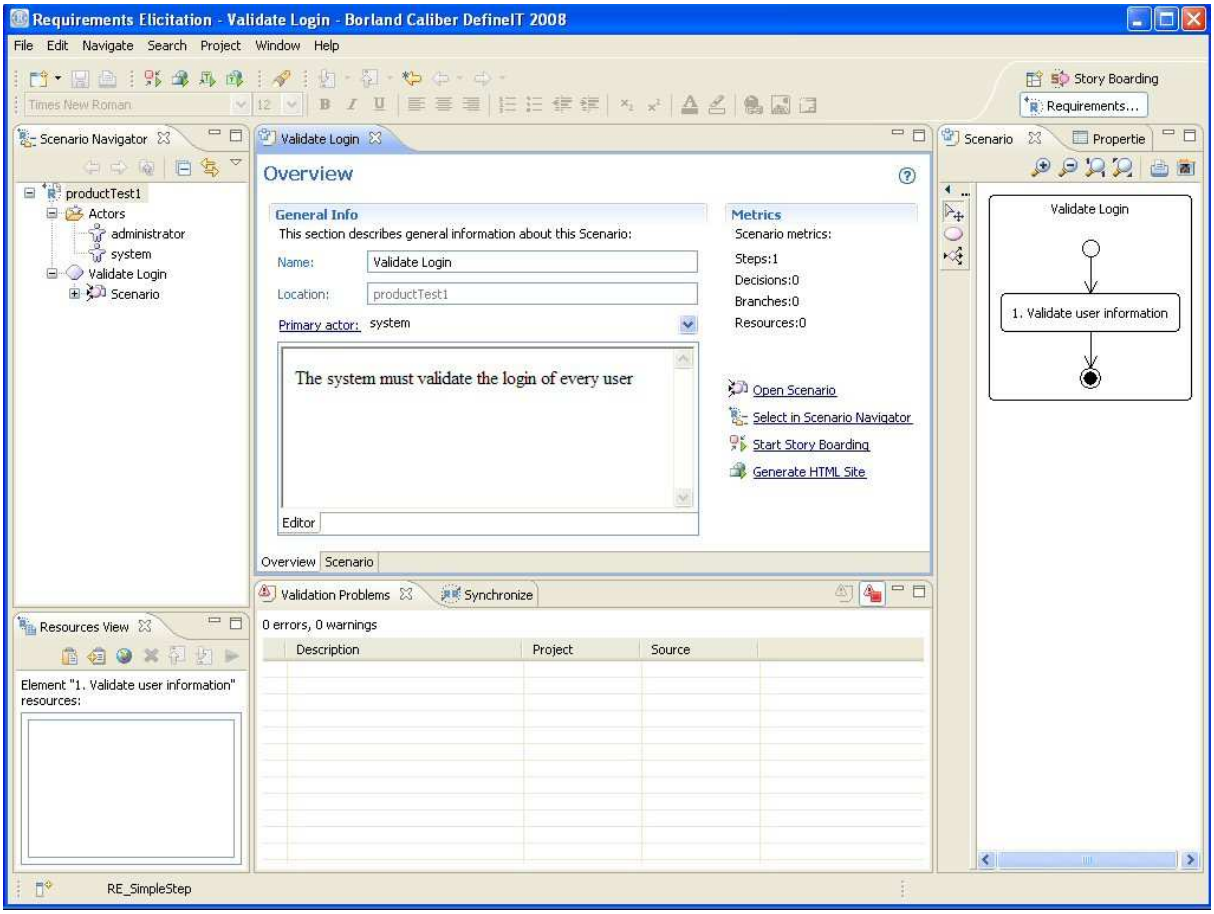

Figure 27: Borland Calibre DefineIT showing textual requirements

Despite the rich environment for allowing domain-oriented information to be noted and recorded, Borland Calibre does not provide a means to use the recorded requirements directly in trying to produce further models during a software development project.

4.4.1.1.3 Visual Paradigm for UML

This tool provides standard UML modelling capabilities, including use case modelling, recording of textual descriptions to provide detail about use cases or problem domain, class modelling, activity charts, etc.

Whereas the tool provides a means to record textual descriptions of the main diagrams, or the problem domain, such descriptions cannot be used directly to build subsequent visual models.

4.4.1.1.4 Telelogic DOORS – System Architect

Telelogic DOORS is a mainly business change management tool allowing the interrogation and alignment of business strategy to IT solutions. The tool aims to enhance communication among stakeholders regarding business processes using visual models. Analysis of business processes is purely done by viewing of the visual models. There is no way to derive those models from any previous textual descriptions of the domain or business context.

4.4.1.1.5 Topcased

This is a UML modelling tool that also provides an integration infrastructure with Eclipse. Like many UML tools, Topcased allows users to build visual UML models and additional textual descriptions to supplement those models. A disjoint though is that there is no way for users to build the visual models directly from the textual descriptions. Hence, any descriptions of the problem domain, can only be referred to rather than used directly in deriving the visual models and these are UML models as opposed to pre-CIM models.

4.4.1.1.6 Telelogic DOORS – Artisan Studio or Doors Analyst

DOORS is a modelling tool that enables analysts to supplement textual requirements with visual models whilst maintaining traceability of the models back to requirements. Textual descriptions of requirements or other information can be automatically updated with the subsequent visual models. This is an interesting feature of Doors that would benefit the scrapbook modeller in terms of traceability.

4.4.1.1.7 OpenAmeos

OpenAmeos is a UML tool that supports diagram interchange via XMI files. It is possible to produce fragments of a problem domain description using XMI, and importing that in OpenAmeos to produce a UML diagram. This is quite restricting because a user has to be conversant with XML, and also has to decide which UML diagram to produce. There is not direct support of the problem domain description creation, nor the use of such description in producing informal visual models of the domain.

4.4.1.1.8 Magic Draw

MagicDraw is a UML modelling tool that can be integrated into Telelogic DOORS, and has pluggable commercial components for requirements modelling and management. Textual requirements descriptions can be exported into a UML use case diagram. There is no support for informal modelling similar to that of the scrapbook or analysis palette.

4.4.1.1.9 Select Solution for MDA

The main component is Select Architect. The MDA component is model transformation tool to generate models and reverse engineer them based on the UML notation set. The vendors claim the support for CIM, PIM and PSM, but there are no trial copies to verify this support.

Additionally, there is no support for documentation of problem domain issues, nor the use of such documentation in subsequent MDA phases.

4.4.1.1.10 Innovator

It is essentially a MDA tool that supports business modelling using the concept of a business use case, and other modelling constructs such as UML activity charts. Whereas there are various features supporting MDA-type of modelling, there is no support for building of descriptions of the problem domain that may subsequently be used to feed into MDA phases.

4.4.1.1.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	Magic Draw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.1.2 Selection of desired scrap text from the domain description to create a visual informal scrap model.

This is an important feature of the domain analysis tool, the Scrapbook modeler in particular. It concerns the ability to import files containing problem domain descriptions, and selection of desired content from the files to create a visual, informal scrap model.

4.4.1.2.1 Rational Requisite Pro

Requisite Pro is a requirements management component of the Rational Suite that supports recording and prioritisation of textual requirements. The tool however does not support the selection of parts of such text to create any subsequent models.

4.4.1.2.2 Borland Caliber DefineIT (BCD)

BCD is a tool with rich features for requirements modelling and allows for the association of stakeholders to requirements. There is no support for selection of such requirements or parts of them to produce visual models of the domain.

4.4.1.2.3 Visual Paradigm for UML

This is a widely used UML modelling tool, but does not provide the feature of selecting descriptions of a problem domain to produce any UML model during standard modelling activities or MDA development.

4.4.1.2.4 Telelogic DOORS – System Architect

The Telelogic family of components is quite a rich UML modelling suite, and widely used in MDA development. Like many UML and MDA supporting tools, Telelogic DOORS does not provide a means to select problem domain descriptions for visual model production during problem domain analysis.

4.4.1.2.5 Topcased

This UML modelling tool is commonly used as an Eclipse plug-in, and supports importing of XMI descriptions to produce UML models. There is no way to select non-XMI descriptions and use it to build informal visual models of the domain.

4.4.1.2.6 Telelogic DOORS – Artisan Studio

This tool does not provide a feature for selecting elements of problem domain description for building an informal visual model.

4.4.1.2.7 OpenAmeos

This tool does not provide a means to annotate visual models with text, nor support production of textual supplements of the visual models. Hence, there is no way to select any text which is then used to produce informal models of the domain.

4.4.1.2.8 Magic Draw

MagicDraw is a mainly UML modelling tool, with no support for selection of problem domain descriptions for automatic creation of informal models of the domain for analysis purposes.

4.4.1.2.9 Select Solution for MDA

This toolset does not provide a feature for problem domain documentation, nor a means to use any such description to build informal, visual models directly.

4.4.1.2.10 Innovator

Innovator does not support the building of descriptions of the problem domain, nor is there any capability to select elements of description in building a visual model for analysis.

4.4.1.2.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	Magic Draw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.1.3 Selection of scrap model elements for further analysis and creation of an informal analysis model.

This feature concerns the function of the domain analysis tool allowing the selection of some elements of the visual scrap model for further analysis and the creation of a domain model, termed the Analysis Palette model.

4.4.1.3.1 Rational Requisite Pro

There is no concept of domain analysis within Requisite Pro, and any associated modelling is typically UML use cases, classes and other UML visual models. In any case, Requisite Pro documents do not feed into any associated models.

4.4.1.3.2 Borland Caliber DefineIT (BCD)

The tool does not support analysis of problem domain models, or their refinement. There is no way to select informal model elements and they cannot be created in the first place.

4.4.1.3.3 Visual Paradigm for UML

The tool does not support creation of scrap models, nor selection of any such model elements to refine them into an analysis model of the domain.

4.4.1.3.4 Telelogic DOORS – System Architect

The tool does not support creation of scrap models, nor selection of any such model elements to refine them into an analysis model of the domain.

4.4.1.3.5 Topcased

Topcased has no support for informal modelling, nor the production of UML models from any other source other than XMI files.

4.4.1.3.6 Telelogic DOORS – Artisan Studio

The tool does not support creation of scrap models, nor selection of any such model elements to refine them into an analysis model of the domain.

4.4.1.3.7 OpenAmeos

The tool does not support creation of scrap models, nor selection of any such model elements to refine them into an analysis model of the domain.

4.4.1.3.8 Magic Draw

The tool does not support creation of scrap models, nor selection of any such model elements to refine them into an analysis model of the domain.

4.4.1.3.9 Select Solution for MDA

The tool does not support creation of scrap models, nor selection of any such model elements to refine them into an analysis model of the domain.

4.4.1.3.10 Innovator

The tool does not support creation of scrap models, nor selection of any such model elements to refine them into an analysis model of the domain.

4.4.1.3.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	Magic Draw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.1.4 Refinement of domain model elements into roles, activities or data objects.

This feature concerns the functionality of the domain analysis tool to enable refinement of elements such as bloops into roles or activities, or in deed, the changing of some activities into one or more activities or data objects. It is a feature that is central to problem domain analysis and comprehension.

4.4.1.4.1 Rational Requisite Pro

There is no concept of roles, activates or data objects and any such refinements are not supported.

4.4.1.4.2 Borland Caliber DefineIT (BCD)

There is no concept of roles, activates or data objects and any such refinements are not supported.

4.4.1.4.3 Visual Paradigm for UML

There is no concept of roles, activates or data objects and any such refinements are not supported.

4.4.1.4.4 Telelogic DOORS – System Architect

There is no concept of roles, activates or data objects and any such refinements are not supported.

4.4.1.4.5 Topcased

There is no concept of roles, activities or data objects and any such refinements are not supported. However, when a user produces an activity chart or BPMN model, some of these concepts might become evident. This however is not within domain modelling or analysis.

4.4.1.4.6 Telelogic DOORS – Artisan Studio

There is no concept of roles, activates or data objects and any such refinements are not supported.

4.4.1.4.7 OpenAmeos

There is no concept of roles, activates or data objects and any such refinements are not supported.

4.4.1.4.8 Magic Draw

There is no concept of roles, activities or data objects and any such refinements are not supported. However, when a user produces an activity chart or BPMN model, some of these concepts might become evident. This however is not within domain modelling or analysis palette levels. A user would need to download a BPMN or UML stencil to produce appropriate models.

h) Select Solution for MDA

There is no concept of roles, activates or data objects and any such refinements are not supported.

i) Innovator

There is no concept of roles, activates or data objects and any such refinements are not supported.

4.4.1.4.9 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	Magic Draw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.1.5 Selection of analysis model elements for use in creating an initial business process model

This feature concerns the functionality of the Domain Analysis Tool to allow selection of desired domain model elements for creation of an initial business process model (using the VCLL component)

4.4.1.5.1 Rational Requisite Pro

Requisite Pro does not support any form of business process modelling, nor the use of any analysis model elements for such business process modelling.

4.4.1.5.2 Borland Caliber DefineIT (BCD)

The closest to a business model is a UML activity diagram, and there is no support for deriving such a model from analysis models. No explicit business process model seems to be supported.

4.4.1.5.3 Visual Paradigm for UML

No known support for using elements of a domain model in building a business process model.

4.4.1.5.4 Telelogic DOORS – System Architect

No known support for using elements of a domain model in building a business process model.

4.4.1.5.5 Topcased

Topcased does support importing of XMI files to create respective visual models. There is a possibility to select elements of an analysis model and use them as targeted process elements based on predefined concepts. However, this is not inbuilt into the tool, and a user would need to write a custom application to format and process the XMI file accordingly

4.4.1.5.6 Telelogic DOORS – Artisan Studio

No known support for using elements of a domain model in building a business process model.

4.4.1.5.7 OpenAmeos

No known support for using elements of a domain model in building a business process model.

4.4.1.5.8 Magic Draw

One can download a BPMN stencil to construct business process models, but there is no automated support for selecting domain models to use in creating the business models.

4.4.1.5.9 Select Solution for MDA

No known support for using elements of a domain model in building a business process model.

4.4.1.5.10 Innovator

No known support for using elements of a domain model in building a business process model and there is no consideration for domain analysis or modelling.

4.4.1.5.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.2 CIM

Table 61:Tool chain summary at the CIM level

Tool Chain	Constituting Tools			
	Pre-CIM	CIM	PIM	Code generation
1 - IBM	Rational Requisite Pro	Websphere Business Modeler	Rational Software Architect	Rational Software Architect
2 - Borland	Borland Caliber DefineIT	Borland Together	Borland Together	Borland Together
3 - Visual Paradigm	Visual Paradigm for UML	Business Process Visual Architect	Visual Paradigm for UML	Visual Paradigm for UML

4 - Telelogic	Telelogic DOORS	Telelogic System Architect	Telelogic Modeler/Rhapsody/Tau	Rhapsody/Tau
5 - Eclipse	Topcased	Eclipse STP BPMN	Topcased /STP SOA	Acceleo
6 - Artisan	Telelogic DOORS	Artisan Studio	Artisan Studio	Artisan Studio
7 - openAmeos	openAmeos or Telelogic DOORS	openAmeos	openAmeos	openAmeos
8 - NoMagic & InteractiveObjects	Magic Draw	Magic Draw	Magic Draw	Arcstyler
9 - Select Business Solutions	Select Solution for MDA	Select Architect	Select Solution for MDA	Select synchronizers
10 - MID	Innovator Business + Innovator Object	Innovator Business + Innovator Object	Innovator Object	Innovator Object

4.4.2.1 Import of un-structured requirements information

4.4.2.1.1 IBM Websphere Business Modeler

In this tool, the first step of work is named “Business Process Discovery”, which can create very simple models, but the tool has no import function of unstructured information.

4.4.2.1.2 Borland Together

No known support for importing un-structured requirements information.

4.4.2.1.3 Visual paradigm Business Process Visual Architect

It supports textual analysis to identify candidate items from a problem statement, and to create actual models from the candidates.

4.4.2.1.4 Telelogic System Architect

Allows for rapid building of data models from scratch, checking for basic referential integrity, but has no explicit import of unstructured requirements information.

4.4.2.1.5 Eclipse STP BPMN

No known support for importing un-structured requirements information.

4.4.2.1.6 Artisan Studio

Provides imports from Rational Rose, Telelogic DOORS, Simulink as well as HOOD bridge.

4.4.2.1.7 openAmeos no explicit support for unstructured requirements information

4.4.2.1.8 MagicDraw

Provides integration of Borland Caliber RM to capture and manage business, technical, functional, and operational requirements.

4.4.2.1.9 Select Architect

This tool supports reverse engineering from Rational Rose and XML Schema into models to aid understanding, but no explicit capturing of the unstructured information is provided.

4.4.2.1.10 MID Innovator Business + Innovator Object

This tool includes templates for the development process and aid installation of requirements, but has no import of the unstructured information.

4.4.2.1.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.2.2 Creation of Business Rules from both data structure and organisational structure model elements in addition to natural language

4.4.2.2.1 IBM Websphere Business Modeler

No known support for creating natural language like business rules.

4.4.2.2.2 Borland Together

No known support for creating natural language like business rules.

4.4.2.2.3 Visual paradigm Business Process Visual Architect

No known support for creating natural language like business rules.

4.4.2.2.4 Telelogic System Architect

No known support for creating natural language like business rules

4.4.2.2.5 Eclipse STP BPMN

No known support for creating natural language like business rules.

4.4.2.2.6 Artisan Studio

No known support for creating natural language like business rules

4.4.2.2.7 openAmeos

No known support for creating natural language like business rules

4.4.2.2.8 MagicDraw

No known support for creating natural language like business rules, but OCL rules to ensure model integrity.

4.4.2.2.9 Select Architect

No known support for creating natural language like business rules

4.4.2.2.10 MID Innovator Business + Innovator Object

No known support for creating natural language like business rules

4.4.2.2.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.2.3 Enhanced business process modelling using extended BPMN

- Creating such things as multi-media artifact information
- Creating data elements, organisational data and business rules inside the process. (These can be used singly or combined.)

4.4.2.3.1 IBM Websphere Business Modeler

Only support for process flows.

4.4.2.3.2 Borland Together

No known support for creating multi-media artefacts or additional elements inside the process view.

4.4.2.3.3 Visual paradigm Business Process Visual Architect

Data objects can appear in every part within a business process.

4.4.2.3.4 Telelogic System Architect

This tool can do it partly, as it is possible to extend Visio diagrams with new constructs, but it is not known if there is a semantics behind those newly created objects.

4.4.2.3.5 Eclipse STP BPMN

One can extend the Intalio BPMN.model in order to extend the palette, but no direct semantical support for it.

4.4.2.3.6 Artisan Studio

With UML and SysML this tool allows for extending the modelling palette, but has no support for BPMN.

4.4.2.3.7 openAmeos

No known support for BPMN, it is a UML modelling tool with support for MDA.

4.4.2.3.8 MagicDraw

A new product Cameo Business Modeler with the support for BPMN 2.0 should be coming, but at the time no support for BPMN.

4.4.2.3.9 Select Architect

No known support for data or organization modelling within BPMN diagrams.

4.4.2.3.10 MID Innovator Business + Innovator Object

UML activity and structure diagrams, but no known support for BPMN.

4.4.2.3.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.2.4 Creation of organisational and data structures using elements helpful to the non-IT person

4.4.2.4.1 IBM Websphere Business Modeler

No known support for creating organisational or data structures.

4.4.2.4.2 Borland Together

Support for data modelling.

4.4.2.4.3 Visual paradigm Business Process Visual Architect

Full support of data and organisational structures.

4.4.2.4.4 Telelogic System Architect

Constructs for modelling data and organization are provided

4.4.2.4.5 Eclipse STP BPMN

Data modelling within BPMN, also associations between data object possible, but no data view per se. Organizational modelling possible through the use of BPMN lanes.

4.4.2.4.6 Artisan Studio

Provides support for UML, SysML, DoDAF and MDA, but no known support for data and organizational modelling except for UML.

4.4.2.4.7 openAmeos

No known support except for UML 2.0 Profiles

4.4.2.4.8 MagicDraw

No known support except for BPMN 2.0 modelling in the coming Cameo Business Modeller.

4.4.2.4.9 Select Architect

No known support for data or organization modelling within BPMN diagrams, but outside of BPMN data Modelling is supported.

4.4.2.4.10 MID Innovator Business + Innovator Object

This feature is partly supported through data modelling.

4.4.2.4.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.2.5 Export of information from the process model as XPDL (for a workflow engine)

4.4.2.5.1 IBM Websphere Business Modeler

It allows enhancing of models with technical details, creation of execution models that will be used by process engines for business process automation.

4.4.2.5.2 Borland Together

It supports the export into BPEL4WS.

4.4.2.5.3 Visual paradigm Business Process Visual Architect

No known support for XPDL export.

4.4.2.5.4 Telelogic System Architect

No known support for XPDL export.

4.4.2.5.5 Eclipse STP BPMN

No known support for exporting XPDL, but provides support for BPEL export.

4.4.2.5.6 Artisan Studio

It provides only XMI 2.1 output and Rational Rose export for model interchange.

4.4.2.5.7 openAmeos

No known support for any type of export formats except for HTML and RTF report generation.

4.4.2.5.8 MagicDraw

Support for transformation to XML Schema, WSDL, EMF UML2, MOF XMI.

4.4.2.5.9 Select Architect

Support for UML XMI serialization.

4.4.2.5.10 MID Innovator Business + Innovator Object

Support for transformation into BPEL4WS

4.4.2.5.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.2.6 Export of a process model as a blueprint to allow creation of UML2 diagrams

4.4.2.6.1 IBM Websphere Business Modeler

No known support for creating UML2 diagrams.

4.4.2.6.2 Borland Together

Support of model transformations.

4.4.2.6.3 Visual paradigm Business Process Visual Architect

No known support for creation of UML2 diagrams.

4.4.2.6.4 Telelogic System Architect

Partial support for MDA in Telelogic Tau and Rhapsody.

4.4.2.6.5 Eclipse STP BPMN

No support for UML diagrams.

4.4.2.6.6 Artisan Studio

The software itself is an UML2 modelling tool with support for MDA open architecture and pattern-based code generation.

4.4.2.6.7 openAmeos

The software itself is an an UML2 modelling tool with support for MDA model transformations

4.4.2.6.8 MagicDraw

UML support and integration with AndroMDA MDD tool.

4.4.2.6.9 Select Architect – UML support and MDA model transformation support

4.4.2.6.10 MID Innovator Business + Innovator Object

Support for UML in Innovator Object.

4.4.2.6.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.3 CIM to PIM Transition

4.4.3.1 Automatic creation of class model from business process model (by use of heuristics for BPM-to-Class mapping).

This feature concerns the application of heuristics on business process models (build using VCLL) to produce class models as an initial design model.

4.4.3.1.1 WebSphere Business Modeller

Websphere supports business process modelling and allows the definition of services. No support is provided for producing class diagrams from business process models

4.4.3.1.2 Borland Together

This Borland tool allows construction of UML class diagrams, but does not support derivation of such diagrams from any other model. There is no concept of heuristics for class model creation either.

4.4.3.1.3 Business Process for Visual Architect

No known support for using elements of a business model to create a class model, nor the use of any heuristics to determine mappings between the two models.

4.4.3.1.4 Telelogic System Architect

No known support for using elements of a business process model to create subsequent class diagrams.

4.4.3.1.5 Topcased

Topcased does support importing of XMI files to create respective visual models. Topcased can be used to create activity charts as part of business modelling, but there is no way to use XML descriptions of such models to produce class models within standard Topcased functionality.

4.4.3.1.6 Artisan Studio

One can build UML class models, no known support for generating class models automatically from business process models

4.4.3.1.7 openAmeos

No known support for using elements of a business process model to create a subsequent class model as initial attempt to design.

4.4.3.1.8 MagicDraw

One can download a BPMN stencil to construct business process models, but there is no automated support for applying heuristics on a BPMN model to produce a class model as initial design model.

4.4.3.1.9 Select Solution for MDA

No known support for producing class models from business process models.

4.4.3.1.10 MID Innovator

Whereas one can build class models from scratch, there is no known support for producing class models from business process models.

4.4.3.1.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	

Complete

4.4.3.2 Automatic creation of an activity chart from a business process model (by use of heuristics for BPM-to-Activity mapping).

This feature concerns the application of heuristics on business process models (built using the VCLL) to produce an activity chart as a behaviour model during design.

4.4.3.2.1 IBM Websphere

Websphere supports business process modelling, and allows definition of services. No support is provided for producing activity charts from business process models.

4.4.3.2.2 Borland Together

This Borland tool allows construction of UML activity charts, but does not support derivation of such models from any other model. There is no concept of heuristics for activity chart creation either.

4.4.3.2.3 Business Process for Visual Architect

No known support for using elements of a business model to create activity chart, nor the use of any heuristics to determine mappings between the two models.

4.4.3.2.4 Telelogic System Architect

No known support for using elements of a business process model to produce activity charts.

4.4.3.2.5 Topcased

Topcased does support importing of XMI files to create respective visual models. Topcased can be used to create activity charts as part of business modelling, but there is no automatic way produce activity charts from business process models using standard functionality provided by Topcased.

4.4.3.2.6 Artisan Studio

One can build UML activity models, but there is no known support for generating activity models automatically from business process models.

4.4.3.2.7 openAmeos

No known support for using elements of a business process model to create an activity chart as part of behaviour modelling at PIM.

4.4.3.2.8 MagicDraw

One can download a BPMN stencil to construct business process models and most of UML diagrams, but there is no automated support for applying heuristics on a BPMN model to produce activity as part of behaviour modelling at PIM level.

4.4.3.2.9 Select Solution for MDA

No known support for producing activity charts from business models.

4.4.3.2.10 MID Innovator

Whereas one can build activity models from scratch, there is no known support for producing such models from business process models.

4.4.3.2.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.4 PIM

Table 62:Tool chain summary PIM level

Tool Chain	**Constituting Tools**			
	Pre-CIM	**CIM**	**PIM**	**Code generation**
1 - IBM	Rational Requisite Pro	Websphere Business Modeler	Rational Software Architect	Rational Software Architect
2 - Borland	Borland Caliber DefineIT	Borland Together	Borland Together	Borland Together
3 - Visual Paradigm	Visual Paradigm for UML	Business Process Visual Architect	Visual Paradigm for UML	Visual Paradigm for UML
4 - Telelogic	Telelogic DOORS	Telelogic System Architect	Telelogic Modeler/Rhapsody/Tau	Rhapsody/Tau
5 - Eclipse	Topcased	Eclipse STP BPMN	Topcased /STP SOA	Acceleo
6 - Artisan	Telelogic DOORS	Artisan Studio	Artisan Studio	Artisan Studio
7 - openAmeos	openAmeos or Telelogic DOORS	openAmeos	openAmeos	openAmeos
8 - NoMagic & InteractiveObjects	Magic Draw	Magic Draw	Magic Draw	Arcstyler
9 - Select Business Solutions	Select Solution for MDA	Select Architect	Select Solution for MDA	Select synchronizers
10 - MID	Innovator Business + Innovator Object	Innovator Business + Innovator Object	Innovator Object	Innovator Object

4.4.4.1 Model execution at the PIM level

4.4.4.1.1 IBM Rational Software Architect

No. Rational Software Architect can only execute OCL expressions on a given model for its verification.

4.4.4.1.2 Borland Together

No model execution feature is present in the tool.

4.4.4.1.3 Visual Paradigm for UML

No PIM execution is available. It is only possible to generate a complete database schema.

4.4.4.1.4 Telelogic Modeller

No model execution feature is present in the tool.

4.4.4.1.5 Topcased

No model execution feature is present in the tool.

4.4.4.1.6 Artisan Studio

No, simulation techniques (animations) are only used for validating models – as shown in Figure 28 .

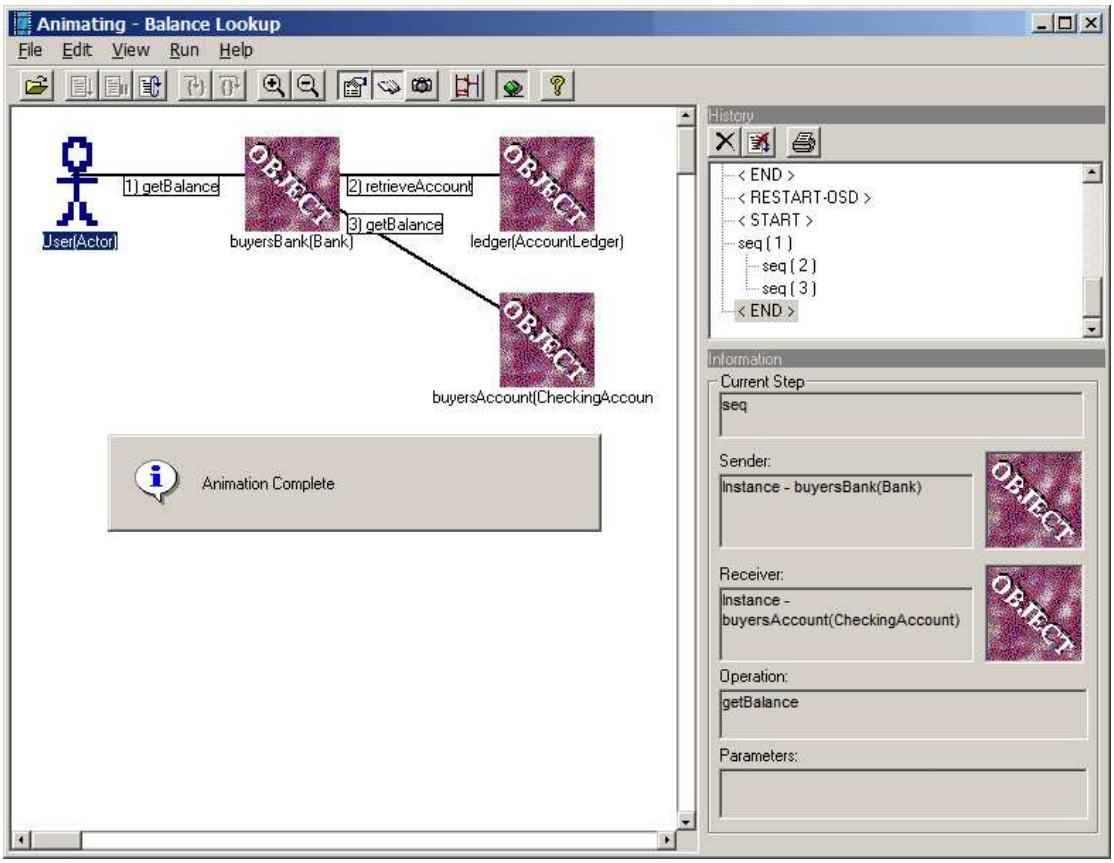

Figure 28: Simulation for model validating in Artisan

4.4.4.1.7 openAmeos

No model execution feature is available in the tool.

4.4.4.1.8 MagicDraw

No model execution feature is available in the tool.

4.4.4.1.9 Select Solution for MDA

(Tool availability issues – we were unable to evaluate this toolchain)

4.4.4.1.10 MID Innovator

No such feature present in the tool.

4.4.4.1.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.4.2 Textual language and editor for UML action modelling

4.4.4.2.1 IBM Rational Software Modeller

No. Rational Software Architect can perform a UML<->Java transformation but only including structural diagrams.

4.4.4.2.2 Borland Together

No textual action modelling feature is present in the tool.

4.4.4.2.3 Visual Paradigm for UML

No textual action modeling feature is present in the tool.

4.4.4.2.4 Telelogic Modeller

No textual modeling of UML Actions present in the tool.

4.4.4.2.5 Topcased

Not present in Topcased. The behaviour body can be attached to an Operation as C, C++, Java, Ada or Python textual code in the form of plain text only as shown in Figure 29.

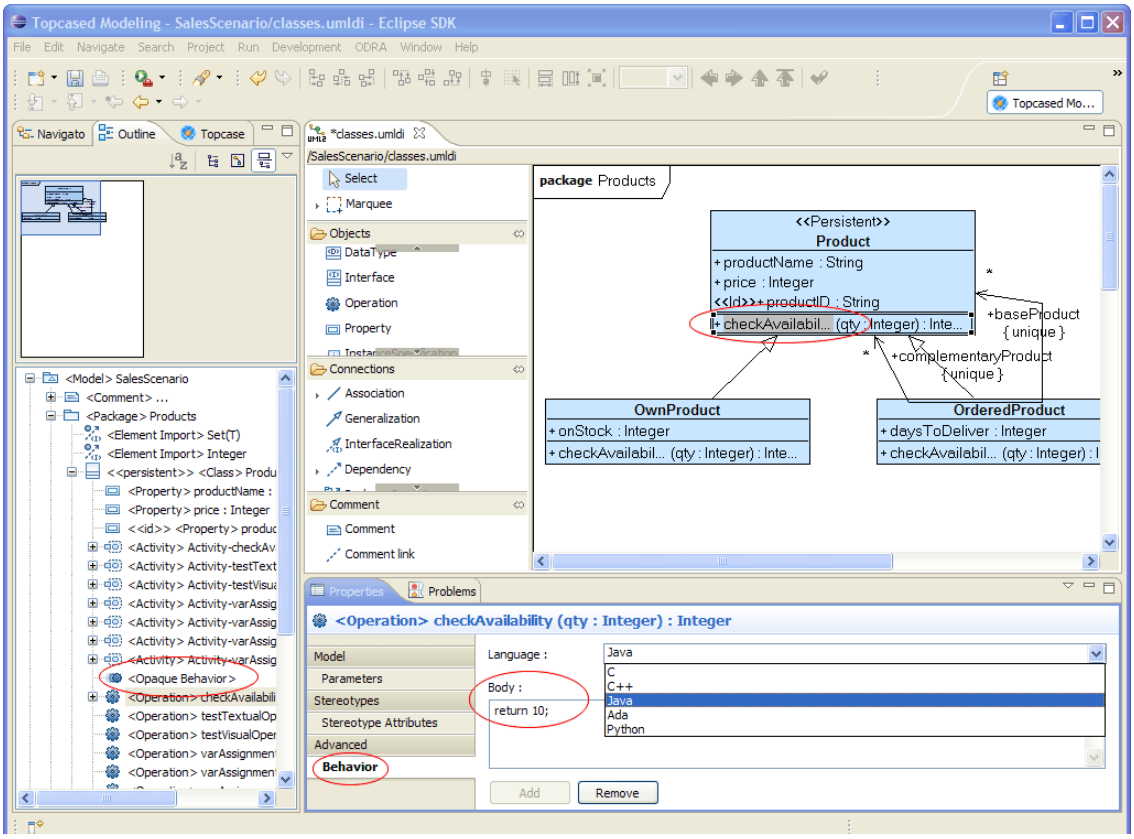

Figure 29: Editing behaviour in Topcased

4.4.4.2.6 Artisan Studio

No, only textual coding is supported for Ada, C++, C, Java and stored as plain text.

4.4.4.2.7 openAmeos

No textual action modelling is supported by the tool.

4.4.4.2.8 MagicDraw

No; although behaviour can be specified in any textual language but there is no dedicated editor and the code is not mapped onto its metamodel instance representations.

4.4.4.2.9 Select Solution for MDA

(Tool availability issues – we were unable to evaluate this toolchain)

4.4.4.2.10 MID Innovator

No such feature present in the tool.

4.4.4.2.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	

Partial	
Complete	

4.4.4.3 Visual language and editor for UML action modelling

4.4.4.3.1 IBM Rational Software Modeller

IBM Rational Software Modeler supports visual action modelling by providing an editor for activity diagrams, shown in Figure 30. Different action types can be selected from a palette.

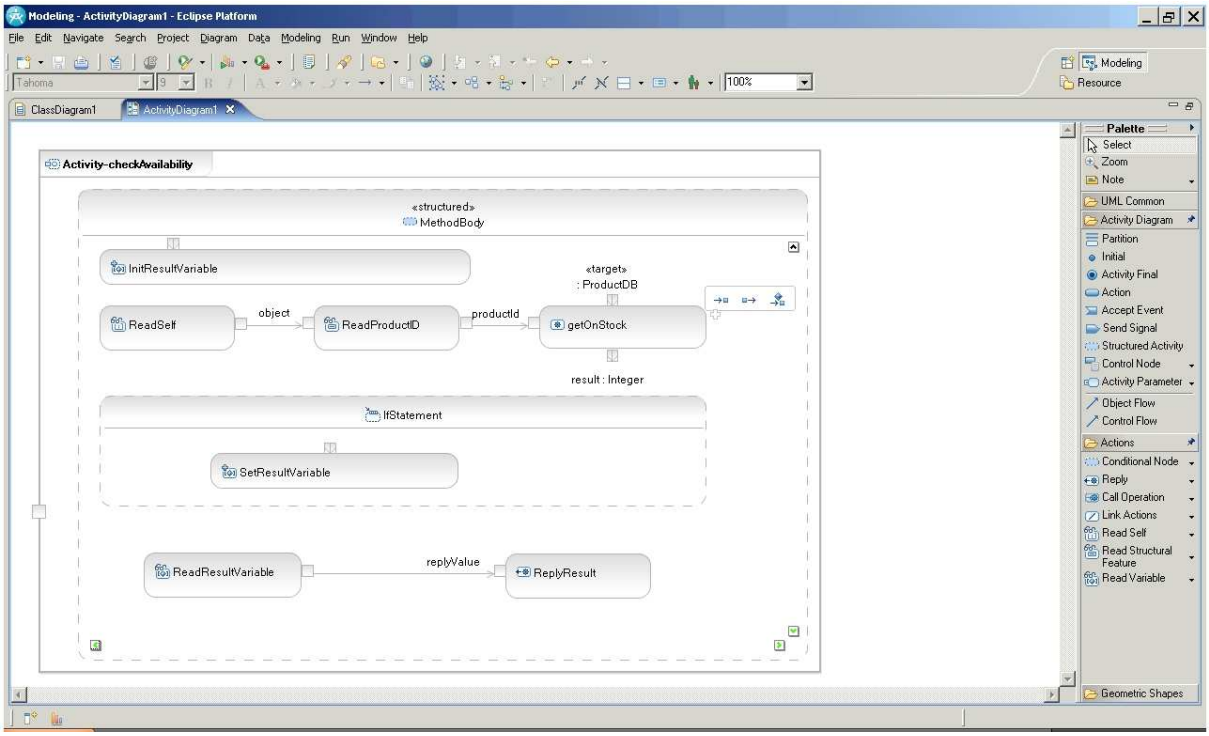

Figure 30: IBM Rational Software Modeler – UML Action modeling

IBM Rational Software Modeler only implements the few action notation proposals contained in the UML specification. Thus all Actions except CallOperationAction look the same, apart from an action-specific symbol left of the action name. Important properties of the actions, e.g. the chosen variable in VariableActions, are not displayed.

Regarding the overall activity notation Variables are not visible, only to be specified via property view of structured nodes. No SequenceNode is available. It is unclear how to specify the test section of conditional nodes.

4.4.4.3.2 Borland Together

Borland Together provides an editor for behavioural modelling in activity diagrams shown in Figure 31. However, there is no way to specify the sub-type of the actions.

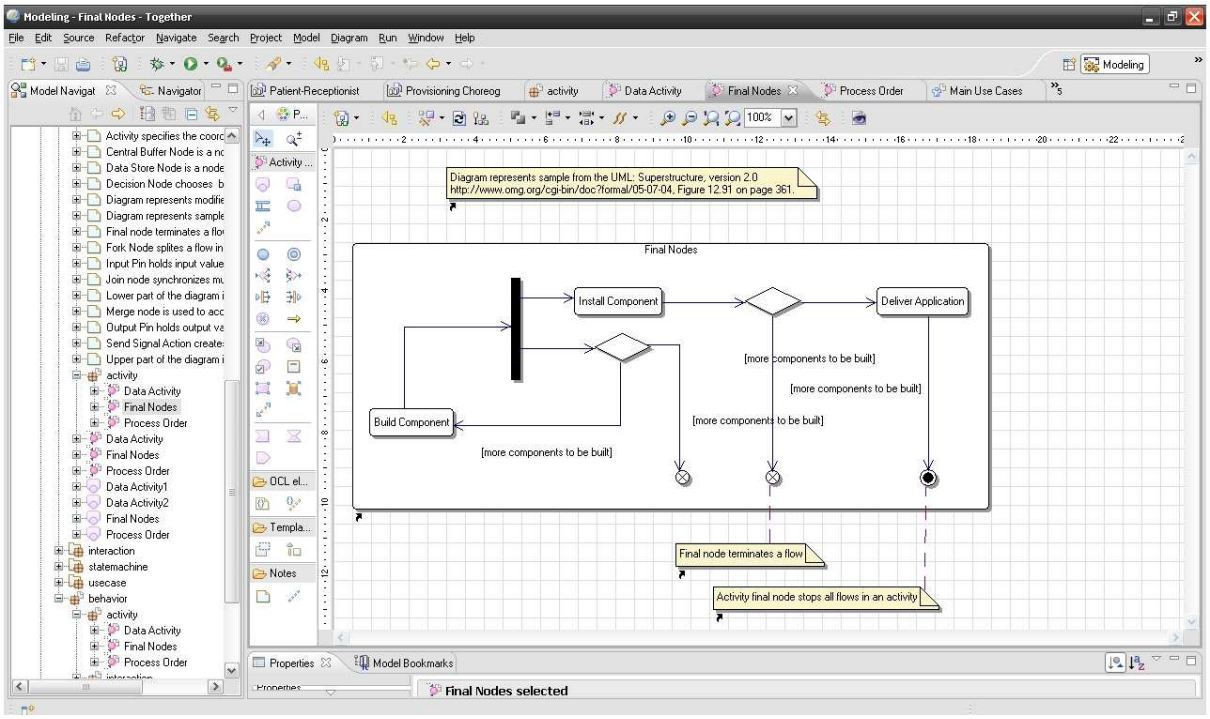

Figure 31: Borland Together – UML action notation

The activity diagram editor of Borland Together only implements the few action notation proposals contained in the UML specification. Thus all Actions look the same. Important properties of the actions, e.g. the chosen variable in VariableActions, are not displayed.

4.4.4.3.3 Visual Paradigm

Visual Paradigm for UML provides an editor for behavioural modelling in activity diagrams shown in Figure 32. However, there is no way to specify the sub-type of the action.

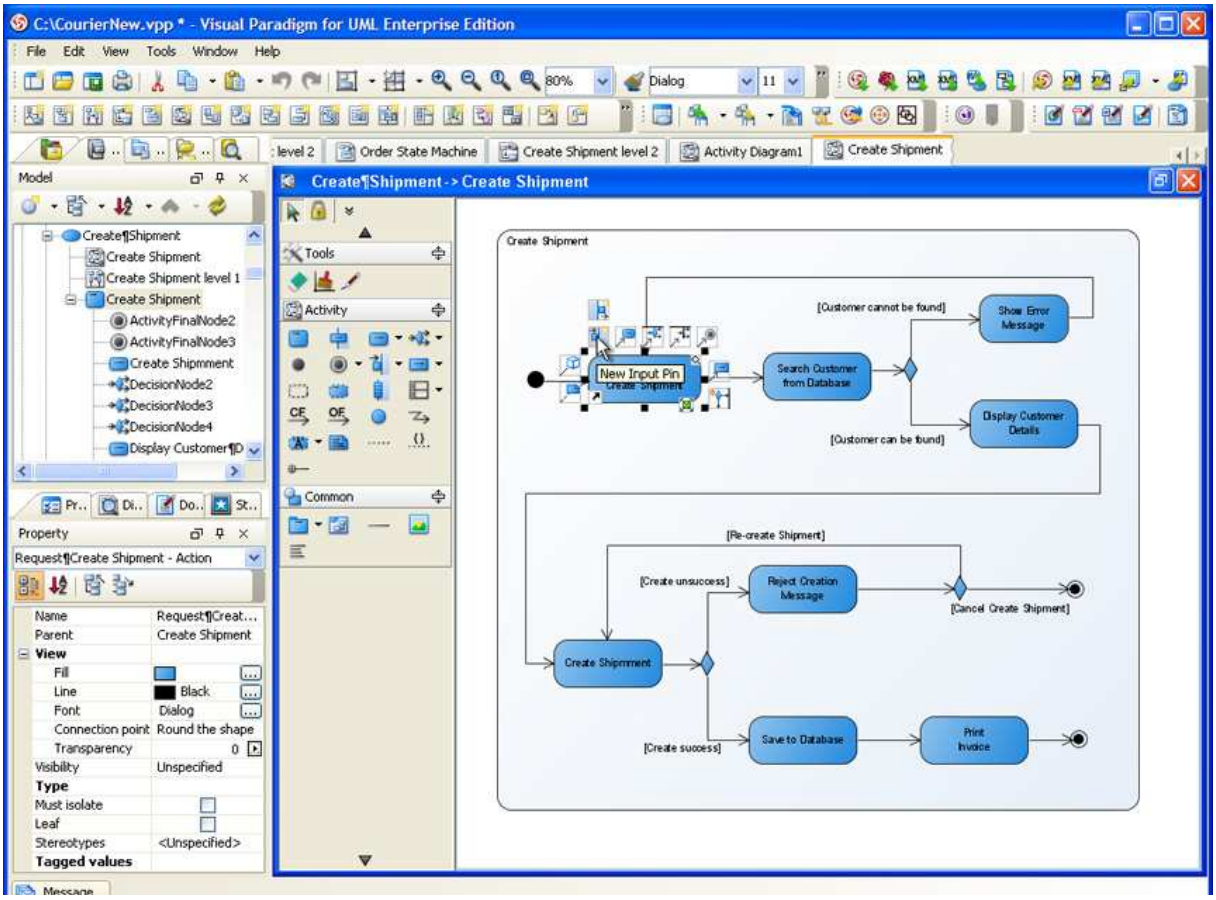

Figure 32: Visual Paradigm for UML – action notation

The activity diagram editor of Visual Paradigm only implements the few action notation proposals contained in the UML specification. Thus all Actions look the same. Important properties of the actions, e.g. the chosen variable in VariableActions, are not displayed.

4.4.4.3.4 Telelogic Modeller

Using Telelogic Rhapsody the system's behaviour can be specified with UML 1.4 Statecharts and SequenceDiagrams.

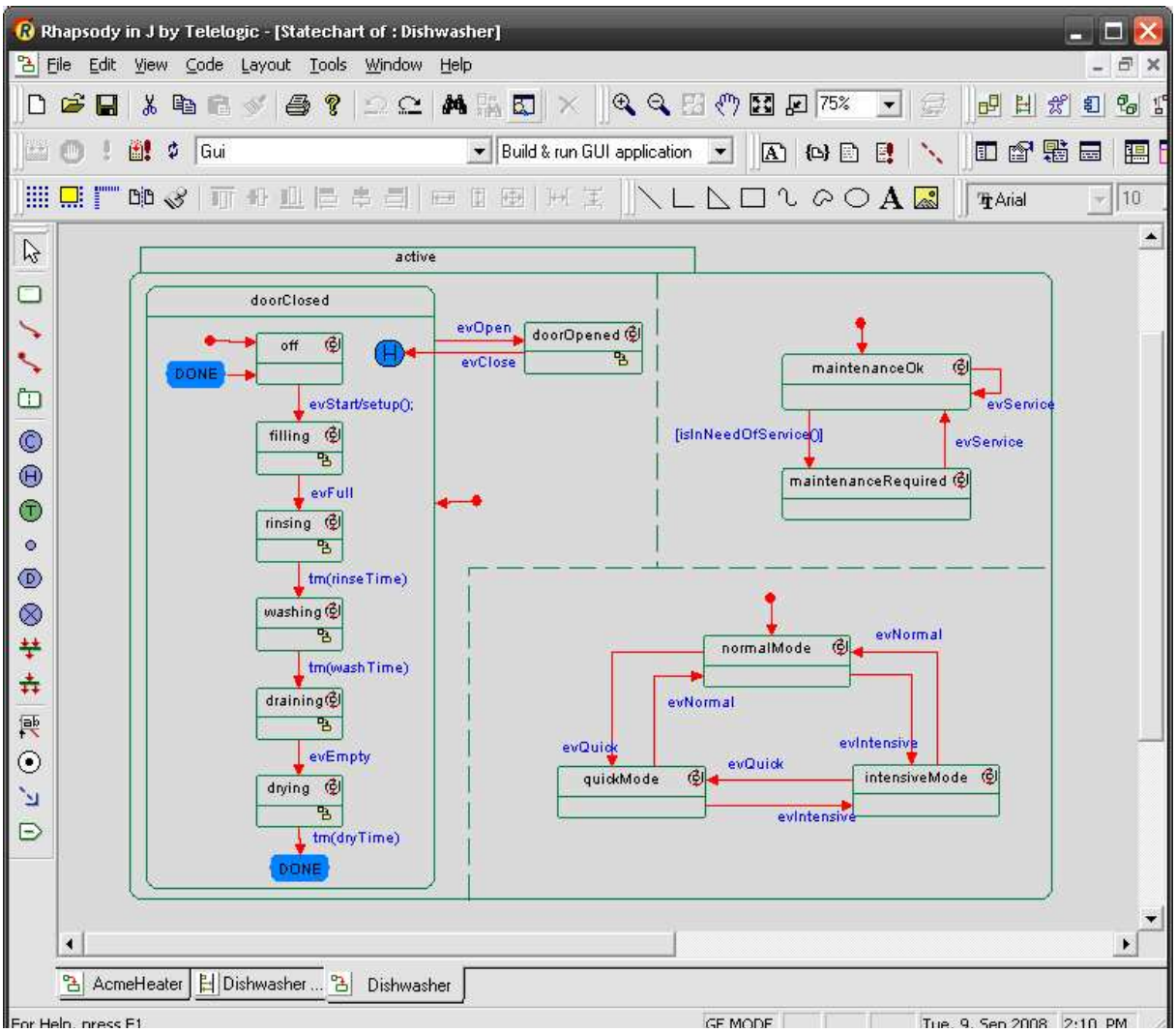

Figure 33: Telelogic Modeler – statechart diagram

However, no fine-grained behavioural modelling using specific actions in activity diagrams is supported. The implementation of the operation's behaviour must be provided in platform-specific code format (Figure 34).

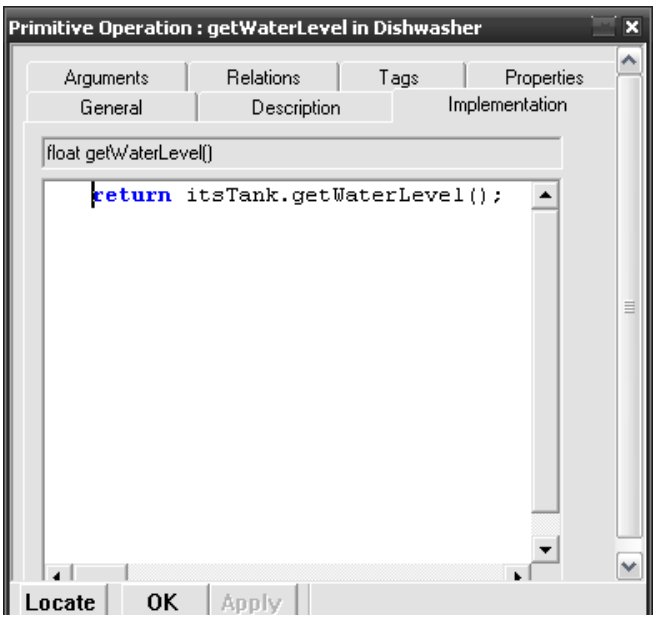

Figure 34: Telelogic Modeler - Operation implementation

4.4.4.3.5 Topcased

Topcased provides support for behavioural modelling with activity diagrams. Different action specifications can be chosen from a palette (Figure 35).

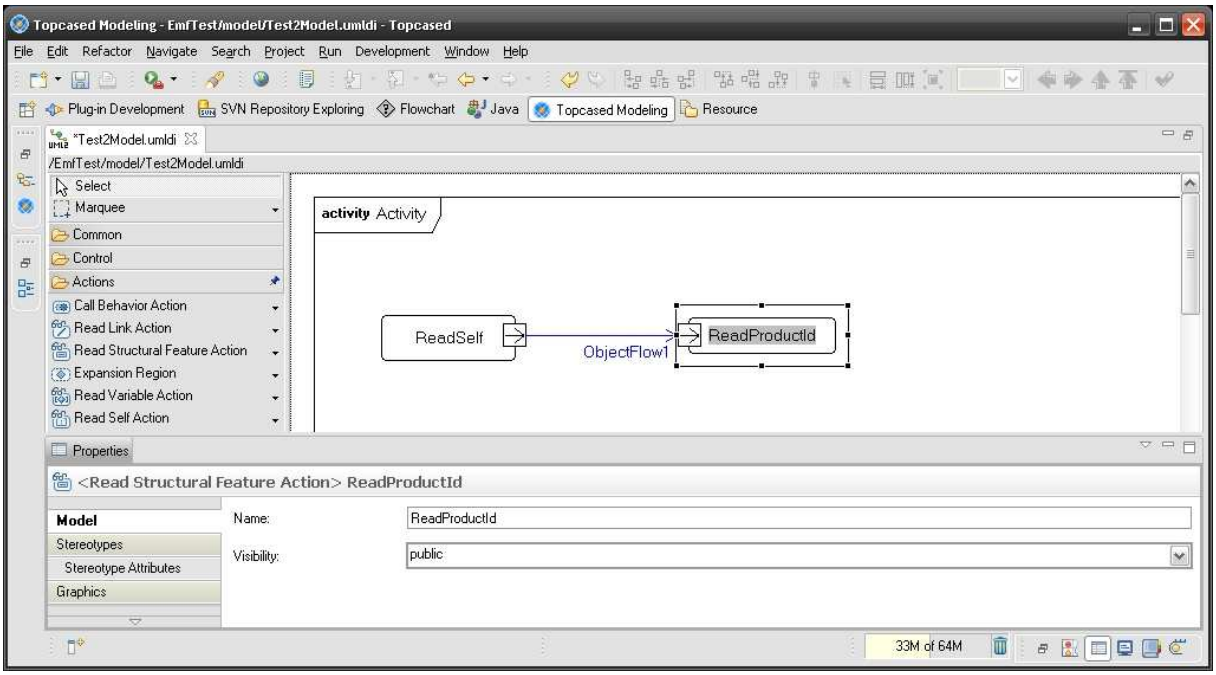

Figure 35: Topcased - Activity diagram

Topcased only implements the few action notation proposals contained in the UML specification. Thus all actions (except CallOperationAction) look the same. Important properties of the actions, e.g. the chosen variable in VariableActions, are not displayed. There is no way to model the body of a sequence node.

4.4.4.3.6 Artisan Studio

Artisan Studio only supports action modelling in UML 1.4 activity diagrams (Figure 36). Specific sub-types of actions can not be selected; there is no visual syntax for specific types of actions

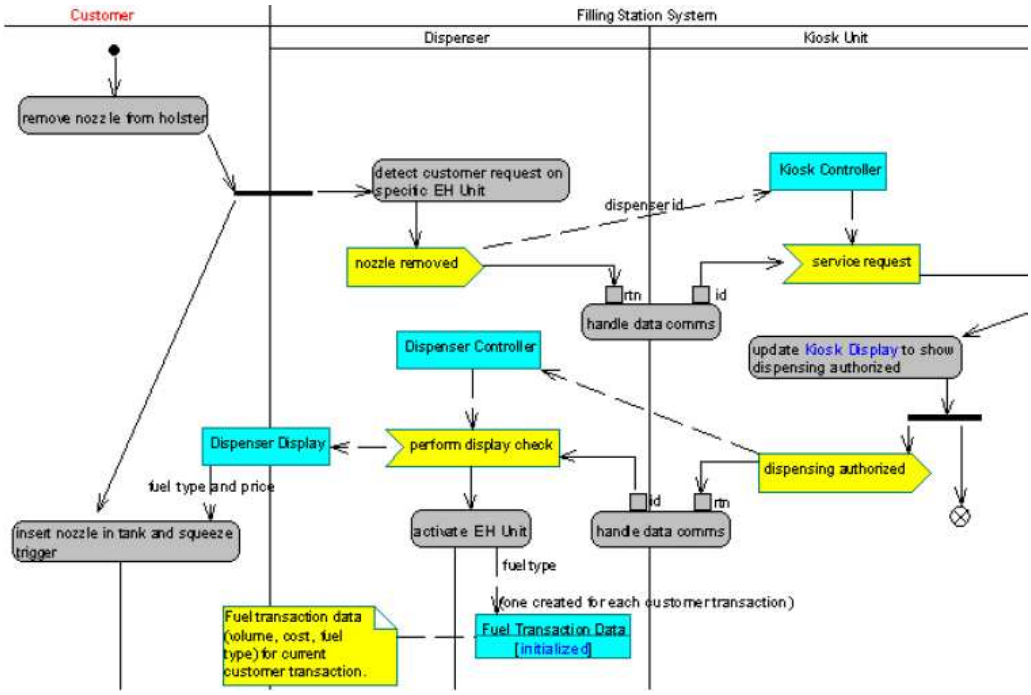

Figure 36: Artisan Studio – Activity diagram

4.4.4.3.7 openAmeos

OpenAmeos only supports action modelling in UML 1.4 activity diagrams (Figure 37). Action subtypes can not be specified; there is no visual notation to clarify the specific task of the actions.

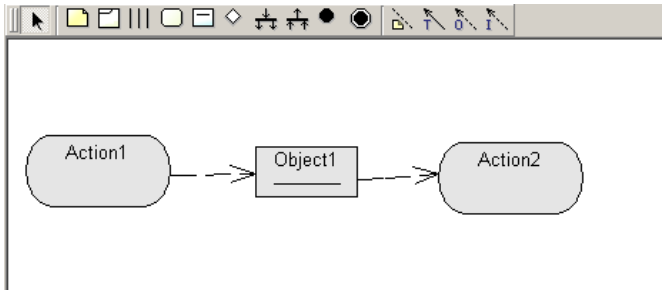

Figure 37: OpenAmeos – Activity diagram

4.4.4.3.8 MagicDraw

MagicDraw supports action modelling in activity diagrams (Figure 38).

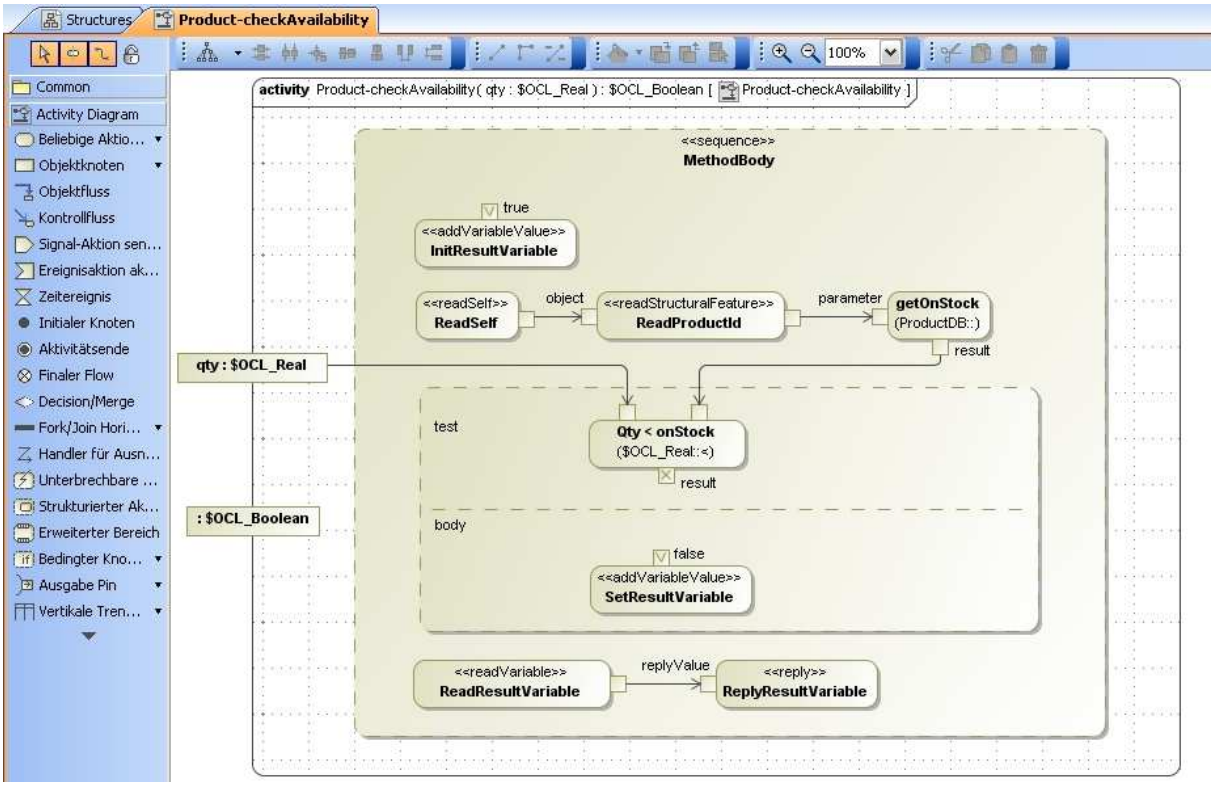

Figure 38: MagicDraw - Activity diagram

MagicDraw only implements the few action notation proposals contained in the UML specification. Thus all Actions (except CallOperationAction) look the same; they are marked by automatically added Stereotypes. Important properties of the actions, e.g. the chosen variable in VariableActions, are not displayed.

4.4.4.3.9 Select Solution for MDA

Select Solution for MDA supports only action modelling in UML 1.4 activity diagrams, as shown in Figure 39.

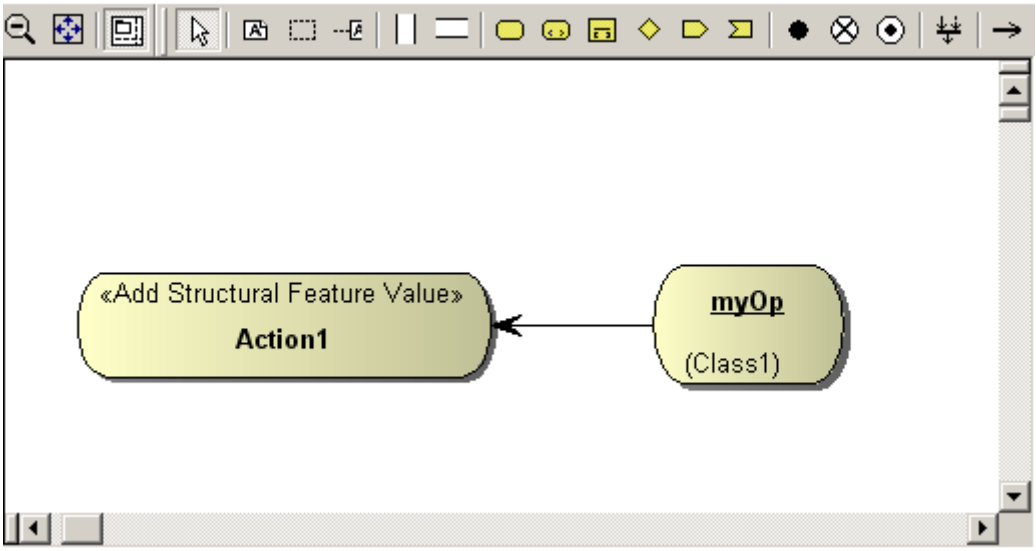

Figure 39: Select Solution – Action modelling

Select Solution does not provide a visual action notation that goes beyond the UML specification (all actions look the same apart from CallOperationAction). The subtype of the action is indicated by a corresponding stereotype. Important details of the actions, e.g. the variable in AddVariableValueAction, are not visualized.

4.4.4.3.10 MID Innovator

MID Innovator only supports UML 1.4 style activity diagrams. As shown in Figure 40, the visualization of the actions can be adapted using icons – however, UML 2.1 action subtypes cannot be specified.

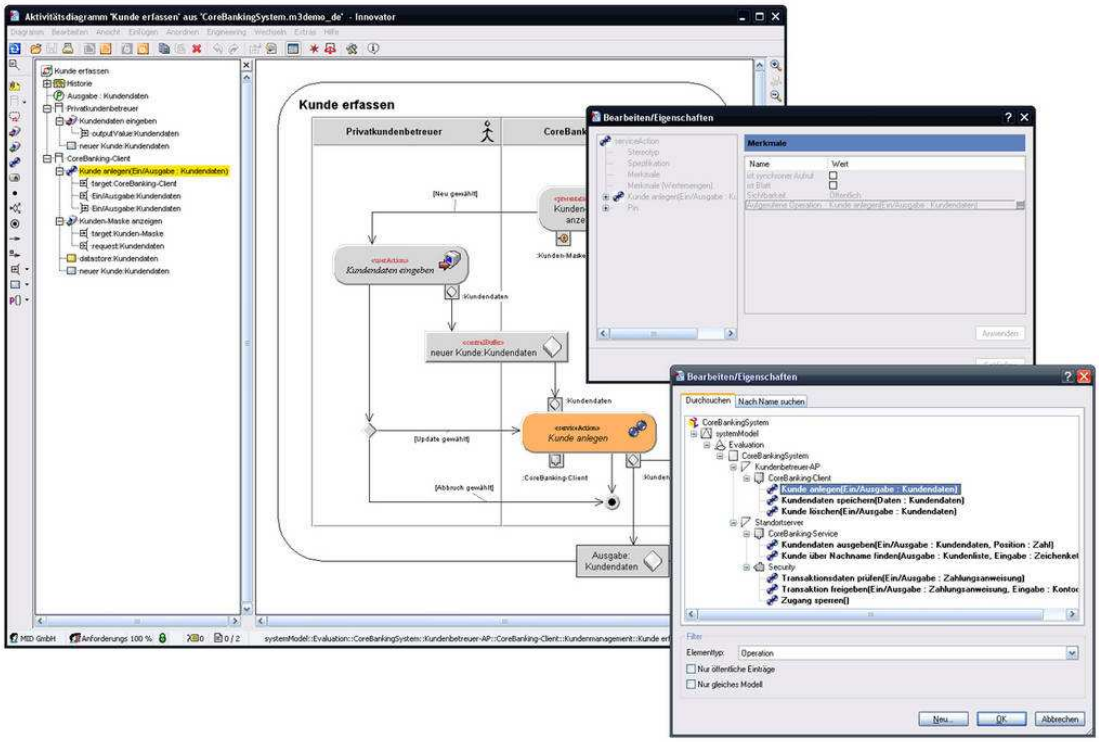

Figure 40: MID Innovator – Activity diagram

4.4.4.3.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.4.4 User-friendly action modelling

4.4.4.4.1 IBM Rational Software Modeller

The behavioural editors of IBM Rational Software Modeler do not help the user by abstracting from complex concepts as e.g. the modelling of pins and flows.

The Editors do not enrich their activity and action notation to provide element-specific information required to understand the model. Thus no pre-selection of important properties is integrated; the editors do not use drop-down lists to support the modelling process. The activity diagram editor has pin error detection. Literal Expressions (e.g. Boolean “true”) are available but are not shown in the editor.

4.4.4.4.2 Borland Together

The behavioural editors of Borland Together do not support modelling of specific actions.

4.4.4.4.3 Visual paradigm for UML

The behavioural editors of Visual Paradigm do not support modelling of specific actions. No additional abstraction is implemented, e.g. hiding the need to deal with pins and flows. The modelling of non-specific actions comes with some helper functionality, e.g. to create a new InputPin, as shown in Figure 32.

4.4.4.4.4 Telelogic Rhapsody

The action modelling of Rhapsody does not support the specification of UML 2.1 action subtypes – thus there is no user support.

4.4.4.4.5 Topcased

Topcased does not help the user by abstracting from complex concepts as e.g. the modelling of pins and flows. There is no wizard and no error checking concerning the input and output pins of actions.

The activity diagram editor does not enrich the activity and action notation to provide element-specific information required to understand the model. Thus pre-selection of important properties and corresponding drop-down lists are integrated only in a separate properties view, not in the model visualization itself.

4.4.4.4.6 Artisan Studio

Artisan Studio does not support UML 2.1 action modelling.

4.4.4.4.7 openAmeos

OpenAmeos does not support UML 2.1 action modelling.

4.4.4.4.8 MagicDraw

MagicDraw does not help the user by abstracting from complex concepts as e.g. the modelling of pins and flows. However, there is action specific support for adding pins, error detection and quick fix in case the user does not model the pins required by the action. Additionally, MagicDraw provides error detection and quick fix in case the number and types of the parameters of the specified method do not comply with the modelled Activity parameters / ActivityParameterNodes.

The activity diagram editor does not enrich the activity and action notation to provide element-specific information required to understand the model. Thus pre-selection of important properties and corresponding drop-down lists are integrated only in a separate properties view, not in the model visualization itself. Variables are not illustrated in the Activities view, they can only be added via context menu.

4.4.4.4.9 Select Solution for MDA

Though the user is able to choose specific actions in the activity diagram view of Select Solution; but there is no way to set the action-specific properties, e.g. the structural feature of an AddStructuralFeatureValueAction. Only UML 1.4 elements are supported, e.g. no StructuredActivityNodes (SequenceNode, ConditionalNode, LoopNode, ExpansionRegion).

4.4.4.4.10 MID Innovator

The visualization of the actions can be adapted using icons – however, there is no support or visualization of UML 2.1 action subtypes.

4.4.4.4.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.4.5 OCL used as a high-level expression and query language

4.4.4.5.1 IBM Rational Software Modeller

No. OCL is only applicable for constraints in the model (*OpaqueExpression* used with *Language*=OCL). OCL expressions may be executed on metamodels.

4.4.4.5.2 Borland Together

OCL is available in the tool only for defining constraints.

4.4.4.5.3 Visual Paradigm for UML

No such feature is present in the tool.

4.4.4.5.4 Telelogic Modeller

No PIM level query language functionality. Only constraints are possible to specify – as plain text.

4.4.4.5.5 Topcased

No OCL or other platform-independent query language is available in the tool.

4.4.4.5.6 Artisan Studio

No OCL queries or other PIM-level query language is present in the tool.

4.4.4.5.7 openAmeos

No OCL queries or other PIM-level query language is present in the tool.

4.4.4.5.8 MagicDraw

Not as a query language. For OCL it is possible to define preconditions and postconditions as constraints for Activities with specifications in OCL. The OCL expressions are textual (they are stored as a plain text) with syntax checking supported – as shown in Figure 44.

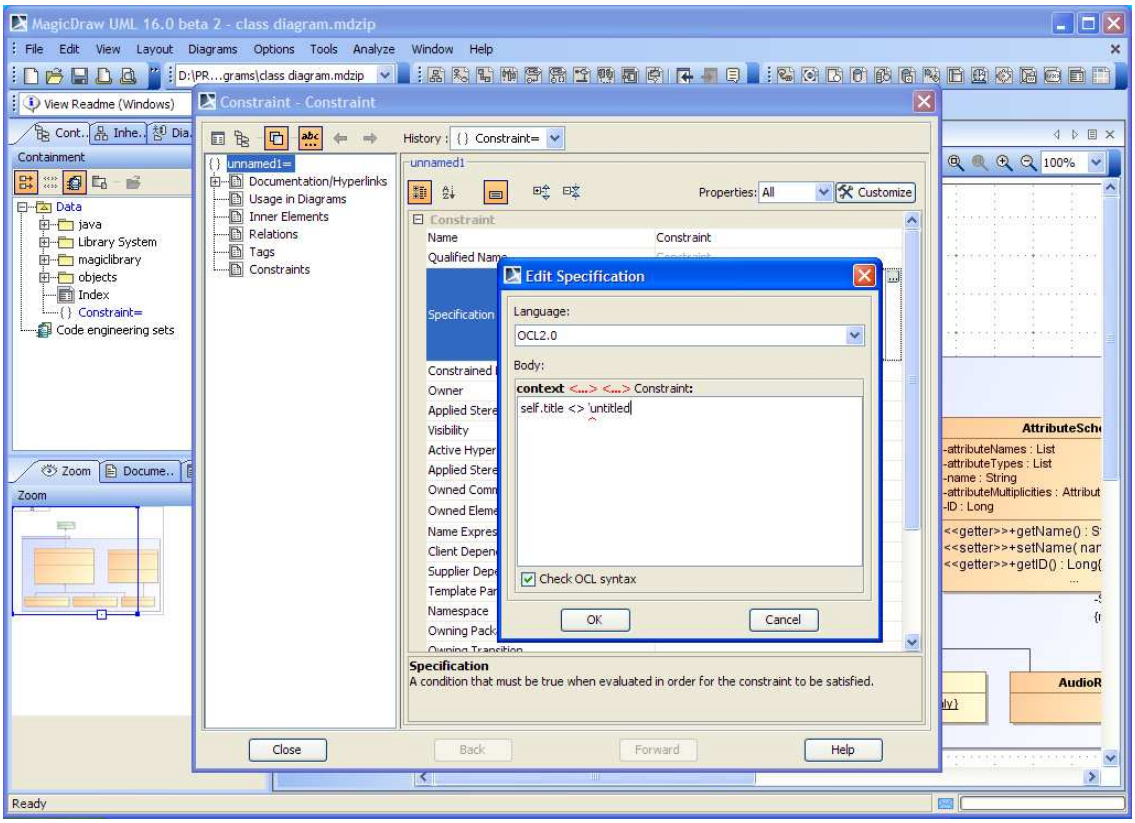

Figure 41: Editing constraints in MagicDraw

4.4.4.5.9 Select Solution for MDA

(Tool availability issues – we were unable to evaluate this toolchain)

4.4.4.5.10 MID Innovator

No such feature present in the tool.

4.4.4.5.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.4.6 OCL query expressions visualized

4.4.4.6.1 IBM Rational Software Modeller

No. Only textual syntax although with context assists.

4.4.4.6.2 Borland Together

No query visualization feature is present in the tool.

4.4.4.6.3 Visual Paradigm for UML

No model level query visualization feature is present in the tool

4.4.4.6.4 Telelogic Modeller

No query visualization feature is present in the tool.

4.4.4.6.5 Topcased

No model-level query visualization is present in the tool

4.4.4.6.6 Artisan Studio

No model level query visualization is supported by the tool.

4.4.4.6.7 openAmeos

No model level query visualization is supported by the tool.

4.4.4.6.8 MagicDraw

No model level query visualization is supported by the tool.

4.4.4.6.9 Select Solution for MDA

(Tool availability issues – we were unable to evaluate this toolchain)

4.4.4.6.10 MID Innovator

No such feature present in the tool.

4.4.4.6.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.4.7 PIM-level aspect-oriented modelling

4.4.4.7.1 IBM Rational Software Modeller

Aspectual FSP Generation Tool

The approach presented by Abeywickrama and Ramakrishnan [2] allows for modelling of crosscutting context-dependent behavior of services as aspect-oriented model. For this purpose, a custom UML profile is used. Model transformations are used to transform the aspects into state machine based behavioral representations. The presented approach bases on the process calculus Finite State Processes (FSP). The corresponding tool called *Aspectual FSP Generation Tool* transforms the modelled aspects into the textual representation of FSP. The aspect and base state machines are then merged using an explicit weaving mechanism at the executable state machine level. The tool is based on the IBM Rational Software Architect and his transformation authoring feature, which is based on Java Emitter Templates (JET).

MATA: Modeling Aspects Using a Transformation Approach

MATA provides capabilities for creating aspect-oriented UML models [3]. The modelling approach can be applied to any modelling language with a well defined metamodel but focus on UML class diagrams, sequence diagrams and state diagrams. The aspect weaving process is considered as a special case of a model transformation. In contrast to our approach, there is no explicit join point model. Each element of the model can serve as a join point.

In MATA, aspects are defined as graph rules. In contrast of all most known approaches, these rules are defined at the level of the concrete syntax/model and not at the meta level. The aspect definition in MATA contains the additional model structure as well as the weaving mechanism/strategy and the definition, where to add the additional structure (pointcut definition). In our approach, the weaving strategy (transformation rules) is defined within the aspect weaver.

The MATA tool works currently on top of IBM's Rational Software Modeler (RSM). The created graph rules are executes using the graph rule execution tool AGG. MATA transforms the created UML model into a instance of a type graph, where the type graph represents a simplified UML2 metamodel. The results are transformed into an RSM compliant representation, so that the results can be displayed in the Rational Software Modeler tool.

4.4.4.7.2 Borland Together

Groher and Schulze presented an approach for modelling aspects ([1]) using the UML lightweight extension mechanisms (UML Profile). The resulting aspect-oriented models can be created and processed using UML compliant tools. In contrast to our approach, no aspect weaving at PIM level is done. The approach proposes and approach for generating AspectJ code. The aspect-oriented concepts are not resolved at the modelling level but are also presented at code level and have to be considered during the PIM to PSM transformation and during the code generation. *Together* from *Borland* was chosen to realize the AspectJ code generator. Since the current version of Borland Together bases on Eclipse, the Eclipse based AOM approaches can also be integrated (see Eclipse section).

4.4.4.7.3 Visual paradigm for UML

There are no Aspect extensions

4.4.4.7.4 Telelogic Modeller

Motorola *WEAVR*

The most known and important AO related extension for Telelogic TAU is the Motorola WEAVR [5]. The Motorola WEAVR allows for modelling crosscutting behaviour based ob state machines in a modular way using aspect-oriented concepts. The add-in for Telelogic TAU provides support for the modularization of crosscutting concerns throughout the system modelling, verification and code-generation phases and provides the following functionality:

- UML 2 Profile that allows users to define aspects
- Join point visualization engine that allows the effects of the modelled aspects to be visualized and validated
- Full aspect weaving at model level
- Simulation engine, that allows the simulation of the aspect models without breaking the modular structure of the aspects

Similar to our approach, an aspect is represented as a stereotyped class. It contains pointcuts and connectors. The connectors, commonly called advice, are connected to the pointcuts using a binds dependency. The pointcuts in the WEAVR approach are modelled using state machines, while our approach provides an own mechanism for constructing pointcut and pointcut expressions.

Furthermore, the WEAVR UML Profile provides capabilities for dealing with the order of precedence of connectors (advice), which are applied to the same join points within an aspect. The scope of the aspect can be restricted statically by using explicit dependencies from the aspect to packages or classes. To allow selection of join points with respect to state machines, the WEAVR provides action based and transition based pointcuts.

Before processing the fully aspect weaving and code generation, the user can simulate the aspect-oriented models.

4.4.4.7.5 Topcased

Execution of aspect-oriented behavioral models using Populo

Fuentes and Sanchez presented an approach for creating and executing aspect-oriented behavioural models [4] based on UML Action Semantics. The additional aspect-oriented constructs are provided using an UML2 Profile. In contrast to our approach, Fuentes and Sanchez use a generic join point model that focuses on message communication between objects. The context exposure for advice constructs is realized by using reflective actions. Our approach uses explicit advice and pointcut signatures that allow for typed context information. Therefore, the information provided by selected join points can be statically validated regarding the parameters expected by advice.

For visual debugging and inspecting the behaviour, the aspect-oriented models can be executed. Before the execution, the aspect-oriented model is woven into a UML conform model, which can be then executed using the Populo virtual machine (http://caosd.lcc.uma.es/populo/). During the weaving process the crosscutting behaviour specified in the aspects is injected into the modules these aspects crosscut as specified by the pointcuts.

The aspect weaving is not required for the further code generation. The code generation process maps the aspect-oriented concepts used in the aspect models into an implementation on an aspect-oriented middleware platform for pervasive systems.

openArchitectureWare:

The openArchitectureWare platform (http://www.openarchitectureware.org/) is an eclipse based MDA framework, which supports the development according to the MDA approach (modeling, model to model transformation, text to model transformation, code generation, textual DSLs, etc.). The openArchitetureWare platform supports aspect-oriented concepts at different levels.

First, oAW allows for non-invasive adaptation of the MDA process. The oAW workflow as well as the transformation (at template level and at Xtend level) can be adapted in a non-invasive way. This kind of adaptation is well suited e.g. for developing different variations of generators and transformations according to the product line engineering.

Second, the XWeave component [7] provides the capability to weave physically a so called aspect model into a base model. The base and aspect model have to conform to the same metamodel. The XWeave weaver can be used to insert additional properties, relationships or additional elements into the base model. The structure of the aspect model is partially used to decide, where to weave. The aspect model can make use of functions, which represents a certain kind of pointcuts. This pointcuts are defined using the oAW expression language.

Third, oAW comes with a join facility called XJoin. XJoin take two or more existing meta models and add relationships joining them.

AMW

The Atlas Model Weaver (http://www.eclipse.org/gmt/amw/) is provided by INRIA as part of the ATLAS Model Management Architecture. The main feature of AMW is to establish links between two or more models or meta models. The result of this process is the weaving model. The building of the weaving model can be done manually or semi-automatically. Based on the weaving model, model transformations can be generated. The linked models can than be merged using the generated transformations.

The weaving metamodel can be extended to add further mapping semantics.

The main areas for using AMW are metamodel comparison, traceability, model matching, model annotation, interoperability. This generic approach can also be used for aspect-oriented modelling (see use case at http://www.eclipse.org/gmt/amw/usecases/AOM/).

M4JPDD

The Join point designation diagrams (JPDD) [6] provide a well designed notation for expressing pointcuts in a visual way. The M4JPDD toolset focus on modelling pointcuts using different concepts of join point selection and permits the developer to model control flow-based, state-based, workflow/dataflow-based, as well as structural join point selections. The modelled pointcuts can be exported in different representations (state charts, AspectJ, JAsCo, etc.)- If the target pointcut language does not contain suitable pointcut designators, the pointcut generator generates additional code to achieve the modelled semantics within the target language. If the JPDD contains for example pattern based pointcut and is exported to AspectJ, the code generator generates a implementation of a state machine within the advice to realize the pointcut semantics without explicit pointcut designators.

4.4.4.7.6 Artisan Studio

There are no Aspect extensions.

4.4.4.7.7 openAmeos

There are no Aspect extensions

4.4.4.7.8 MagicDraw

There are no Aspect extensions

4.4.4.7.9 Select Solution for MDA

There are no Aspect extensions

4.4.4.7.10 MID Innovator

There are no Aspect extensions

4.4.4.7.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.4.8 PIM-level quality defect detection

The PIM-level quality defect detection can be refined into the following detailed list of features provided by VIDE Defect Detector and having only a very basic support among the analysed tool chains.

Table 63: Detailed defect detection feature evaluation against the tool chains analysed

Features \ Tool	IBM Rational Software Architect	Borland Together	Visual Paradigm Suite & Visual Paradigm for UML	Telelogic Modeler & Rhapsody & Tau	Eclipse SOA Tools Platform (STP) Project & Topcased	Artisan Studio	Open Ameos	Magic Draw	Select Solution for MDA	MID Innovator Object Edition
Model Analysis										
Support for single-characteristic* defects	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Support for multi-characteristic defects*	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Use of model metrics	✓	✓	✗	✗	✗	✓	✗	✓	✗	✗
Use of model characteristics	✓	✓	✗	✓	✗	✓	✗	✗	✗	✗
Defect Diagnosis										
Support for single-location defects	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Support for multi-location defects	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
Support to persist Ignore decisions	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Support to (de-) emphasize defect priority	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Support to remove individual defects	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
Support to annotate context factors	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Defect Annotation/Storage:										
Annotation of all UML elements	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Use of EAnnotations within a diagram	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Defect Presentation:										
Use of eclipse marker, eclipse decorators, and icons within a diagram	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Defect differentiation by Type using different Icons	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Defect Differentiation by Priority using colour	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Support for prioritization of defects (based on QM)	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Support for nested defects (e.g., at Comments or Parameter	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

*Regarding metrics we found several tools that use single metrics to assess the model quality, but none of them supported more complex defects comprising different mutually related metrics.

4.4.4.8.1 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.4.9 PIM-level publication of web services - modelling and execution

4.4.4.9.1 IBM Rational Software Architect

IBM Rational Software Architect provides support for web application development targeting JEE 5 and EJB. Web services can be modelled and exported as WSDL document.

4.4.4.9.2 Borland Together

Borland Together allows J2EE modeling (Figure 42) and both export of the WSDL interface and deployment of web application implementations as EAR archives.

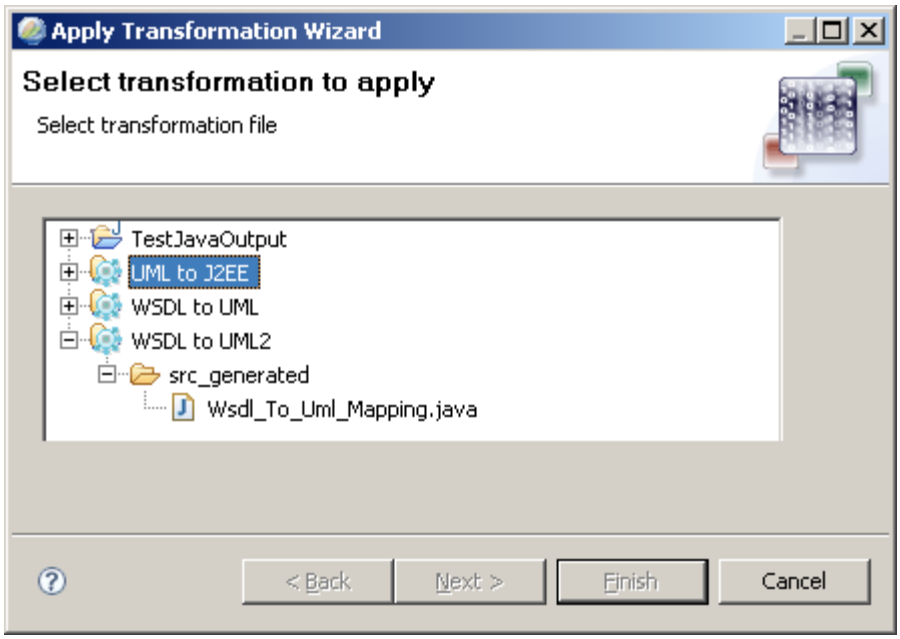

Figure 42: Borland Together – Web service support

4.4.4.9.3 Visual paradigm for UML

Visual Paradigm supports the generation and deployment of web applications, as shown in Figure 43.

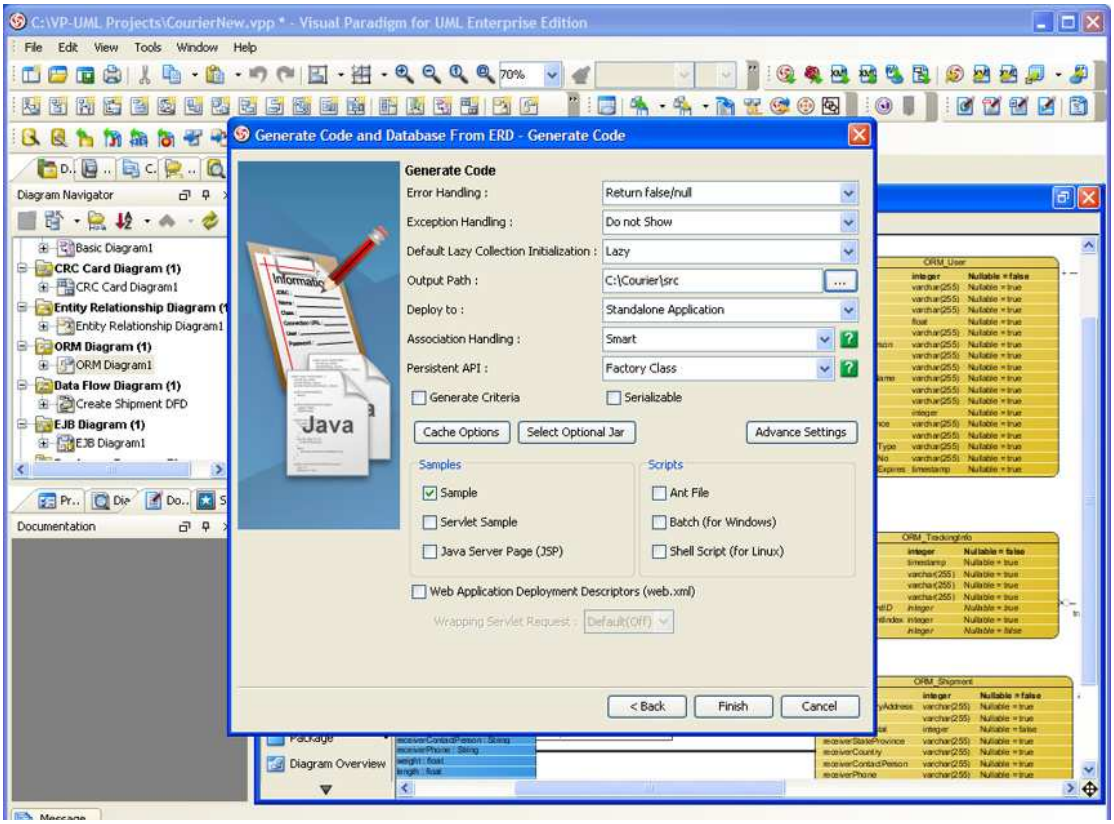

Figure 43: Visual Paradigm – Web application deployment

4.4.4.9.4 Telelogic Rhapsody

Telelogic Rhapsody supports the generation of WSDL (Figure 44). No support for platform-specific web service implementation.

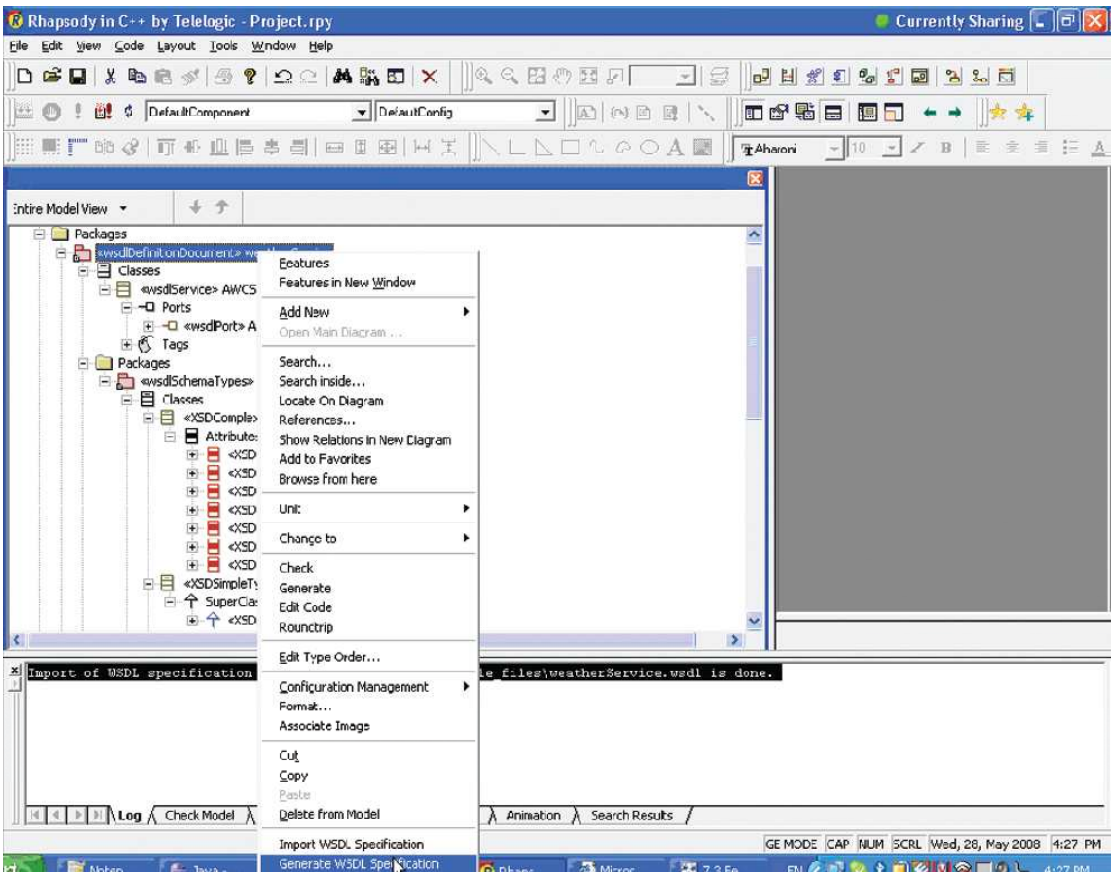

Figure 44: Telelogic Rhapsody – WSDL generation

4.4.4.9.5 Topcased

No pre-defined support for the publication of web services could be found.

4.4.4.9.6 Artisan Studio

No pre-defined support for the publication of web services could be found.

4.4.4.9.7 openAmeos

OpenAmeos has no pre-defined support for the publication of web services.

4.4.4.9.8 ArcStyler

Arcstyler comes with a set of cartridges for J2EE 1.4 and .NET development, including the generation and deployment of EJB 2.1 web service (Figure 45). Currently there is no cartridge for JEE5 / JAX-WS.

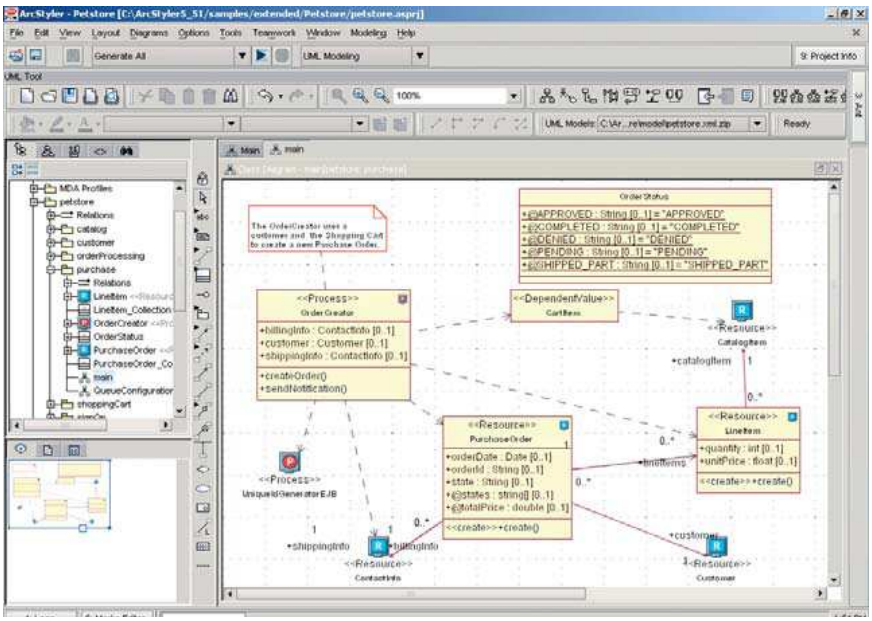

Figure 45: ArcStyler – Web application

4.4.4.9.9 Select Solution for MDA

Select Solution Factory includes support for WSDL, the development and deployment of web services.

4.4.4.9.10 MID Innovator

Web services can be modelled and exported to BPEL code. Additionally, Innovator comes with a set of profiles to support the modelling of web applications (Figure 46). However, the predefined cartridges do not produce web application output.

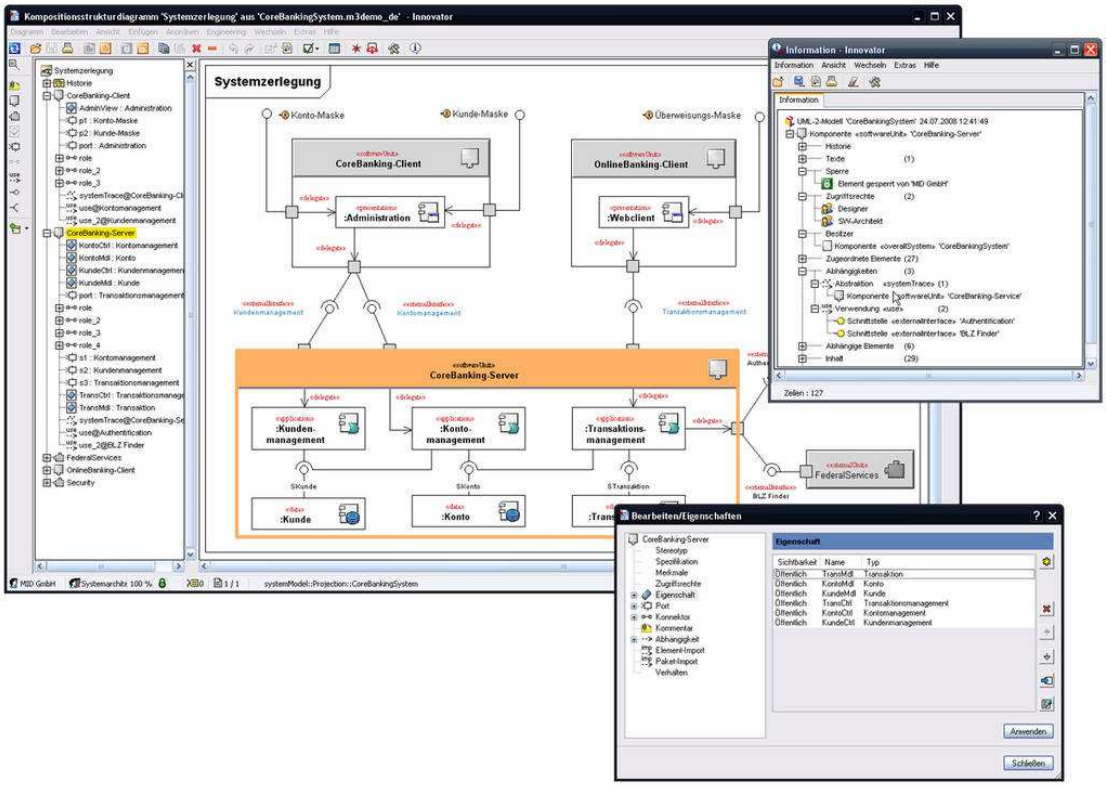

Figure 46: MID Innovator – Web application

4.4.4.9.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.4.10 PIM-level consumption of web services - modelling and execution

4.4.4.10.1 IBM Rational Software Modeller

IBM Rational Software Architect provides support for web application development targeting JEE 5 and EJB. However, a WSDL import could not be found.

4.4.4.10.2 Borland Together

Borland Together comes with a QVT transformation project that converts a WSDL file into a UML model.

4.4.4.10.3 Visual paradigm for UML

No WSDL importer could be detected to support the consumption of web services on PIM level.

4.4.4.10.4 Telelogic Rhapsody

The Net-centric systems pack, a Telelogic Rhapsody add-on, provides the ability to import WSDL files is provided as part of the design model. Moreover, there is round-trip engineering support for Web services interfaces (WSDL). No support for platform-specific proxy class generation

4.4.4.10.5 Topcased

No pre-defined support for the consumption of web services could be found.

4.4.4.10.6 Artisan Studio

No pre-defined support for the consumption of web services could be found

4.4.4.10.7 openAmeos

OpenAmeos has no pre-defined support for the consumption of web services.

4.4.4.10.8 Arcstyler

The Arcstyler cartridges allow the generation of complete web applications. MagicDraw supports import and export of XMI 1.2 documents, enabling the full roundtrip for e.g. WSDL definitions.

4.4.4.10.9 Select Solution for MDA

Select Solution Factory can be used to build complete web applications, especially for the .NET technology platform.

4.4.4.10.10 MID Innovator

No WSDL import could be detected.

4.4.4.10.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.4.11 Platform independent means for GUI development

Graphical user interfaces implemented with WebFace can be published as Java applications or as Java Applets in a web browser. As a runtime environment, they only need a Java Runtime Environment, which is available for many operating systems (for example Windows, Linux,

Solaris, Mac OS X). The same GUI can be used as an Applet or as an application without change, the GUI must only be published again.

Each tool chain consists out of tools for Pre-CIM, CIM, PIM and code generation.

4.4.4.11.1 IBM Rational Software Modeller

This tool chain provides tools for the platform independent development of a GUI, because Java is used as a target platform and there are UI components which can be arranged by drag&drop.

4.4.4.11.2 Borland Together

Borland Together includes a GUI designer. The generated graphical platform independent user interfaces are based on Java.

4.4.4.11.3 Visual paradigm for UML

Visual Paradigm does not provide a designer for graphical user interfaces. If a GUI is needed, a designer of a supported external IDE is used.

4.4.4.11.4 Telelogic Modeller

Rhapsody Graphical Panels provides the possibility to create mock-ups or prototypes of the graphical user interface based on Java

4.4.4.11.5 Eclipse

The development of graphical user interfaces in Eclipse is supported by other Eclipse plug-ins (for example Jigloo, Visual Editor, Window Builder).

4.4.4.11.6 Artisan Studio

Artisan Studio provides no integrated GUI builder.

4.4.4.11.7 openAmeos

OpenAmeos provides no integrated GUI builder.

4.4.4.11.8 MagicDraw

There is no integrated GUI builder.

4.4.4.11.9 Select Solution for MDA

There is no integrated GUI builder.

4.4.4.11.10 MID Innovator

There is no integrated GUI builder.

4.4.4.11.11 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.4.12 Availability of a logical structure of GUI

In contrast to other available Java Development Environments (for example Netbeans), a GUI developed with TNM:WebFace provides an interface to the internal logical connections between the various graphical components. Traditional Java development environments implement the connections between the components with generated Java code. In a GUI generated with WebFace, these connections are implemented as a logical data structure, no code is generated in order to connect the components. This functionality can be used to implement complex help systems. During runtime, the help system can easily monitor the actions of the user by analyzing the events between the components. For example, in the research project Smartkom (www.smartkom.org), the graphical user interface was developed on base of TNM:WebFace. In this project, the data provided by the GUI was used to implement a graphical presentation agent, who guides the users through the running application.

In none of the tools mentioned above, does such a logical structure exist. In all cases, the tools generate Java code which implemented the event listeners of the graphical components. Because the connections are only implemented in the Java code, they are not directly accessible by another process.

4.4.4.12.1 Feature Summary

IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.5 PIM to PSM

4.4.5.1 Code generation of behaviour from PIM (method/constructor bodies)

4.4.5.1.1 IBM Rational Software Modeller

No code generation of behaviour is provided by this tool.

Nevertheless BLU AGE ‘build’ from IBM provides some limited code generation for the behaviour.

BLU AGE ‘build’ is as described from their web site :

BLU AGE ‘build’ is an integrated modelling environment built in Eclipse for UML models elaboration. It offers business application's source code transformation and generation features. This module is designed for consultants and developers who use it on their workstations, set around a UML source repository, if any.

The source code is automatically generated in an Eclipse project form, allowing the applications' execution in the same environment in order to validate their compliance to modelled functional specifications described by UML diagrams.

Figure 47 shows an example of activity diagram that is used to generated code. Note that MagicDraw modelling tool is used, because BLU AGE 'build' is mainly a generator tool. Taking a look at the partition name, we can see 'external screens', 'server', 'screen' and 'reports' which are concepts at the PSM level, not at the PIM level as VIDE language. Therefore we can conclude that, despite this tool generate behaviour, it is not comparable to VIDE model compiler.

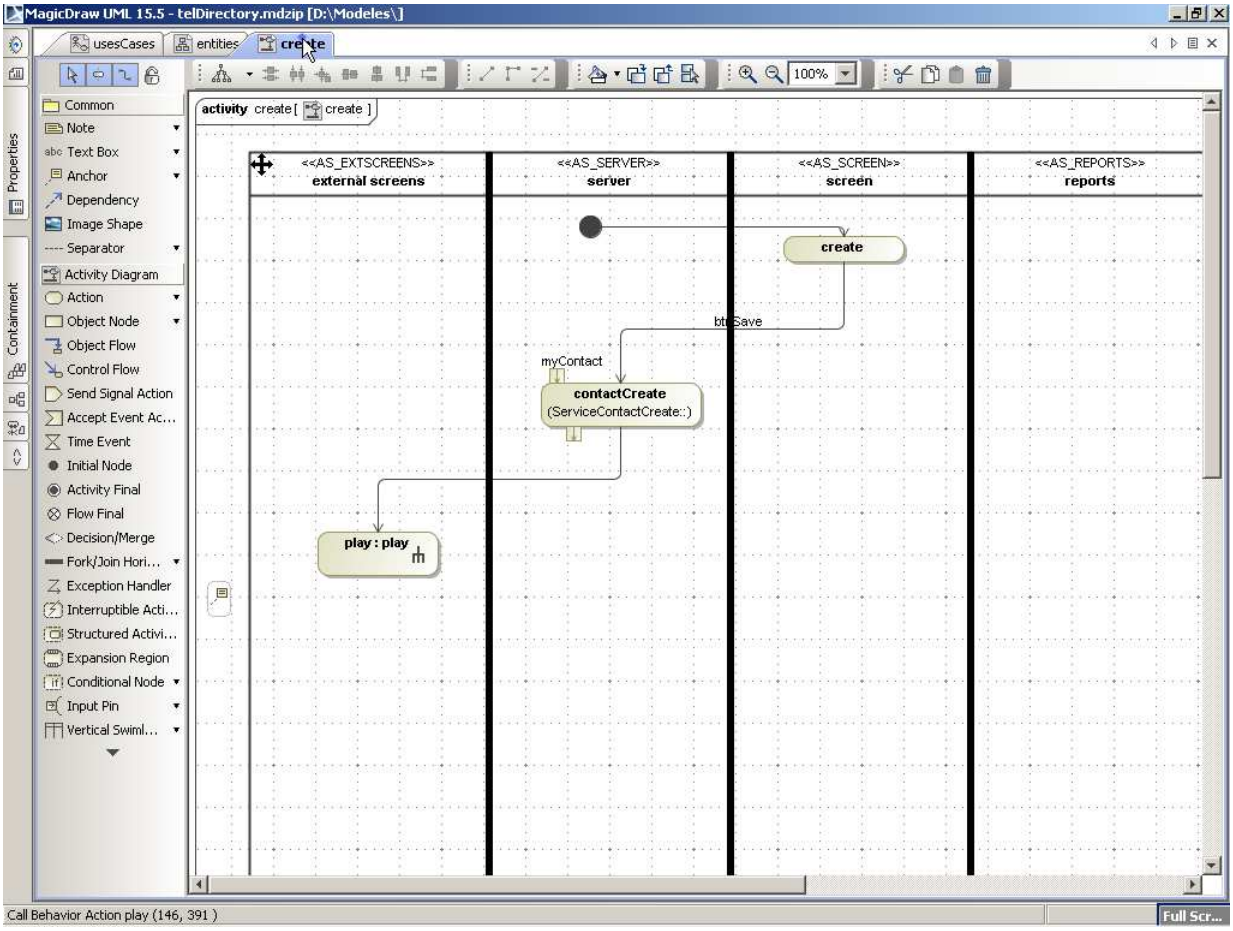

Figure 47: BLU AGE code generation from activity diagram

4.4.5.1.2 Borland Together

Not supported.

4.4.5.1.3 Visual paradigm for UML

Behavioural code generation based on actions is only available for general actions and call behaviour actions in the context of State Machines. There is no mapping for other specific actions modelled in Activity diagrams

4.4.5.1.4 Telelogic Modeller

This tool generates code only for the UML static class diagram. It can also generate sequence diagram from code execution but this is not our purpose.

Telelogic Rhapsody generates code from behavioural model views to C, C++, Java, Ada code. The pre-defined mappings of actions in activity diagrams however do not take into account the specific UML action sub-types.

4.4.5.1.5 Eclipse

Acceleo modules are available for e.g. JEE and C#. These use annotated class diagrams to generate code – a mapping for behaviour modelled as specific actions in activity diagrams is not provided.

4.4.5.1.6 Artisan Studio

Artisan Studio includes the Automatic Code Synchronizer targeting C, C++, C#, Java, Ada and Spark Ada, and a Template Development Kit that allows the customization of the mapping. Though behavioural code can be generated from UML 1.4 behaviour diagrams, Artisan Studio does not provide the possibility to platform-independently specify a method using UML 2 actions and generate code from it.

4.4.5.1.7 openAmeos

This tool generates code only for the UML static class diagram.

4.4.5.1.8 MagicDraw & Arcstyler

The Arcstyler cartridges can only handle behavioural information from UML 1.4 diagrams, specific actions in UML 2.1 activity diagrams are not supported. This tool can also generate sequence diagrams from code execution but this is not our purpose.

4.4.5.1.9 Select Solution for MDA

This tool provides some features to generate code Java code (also C# and C++) from the UML static class diagram and also from predefined technical design patterns and typical architectures (3 tiers, client/servers) but this is not at the level of PIM and it generates only skeletons.

4.4.5.1.10 MID Innovator

This tool provides some features to generate code for behaviour based on the Nassi-Shneiderman diagram notation. The figure below shows an example for such a diagram at a high level of abstraction. The main difference with VIDE language are:

- ☞ Atomic instruction are expressed in target language, C/C++ and Cobol
- ☞ There is no integration with UML models

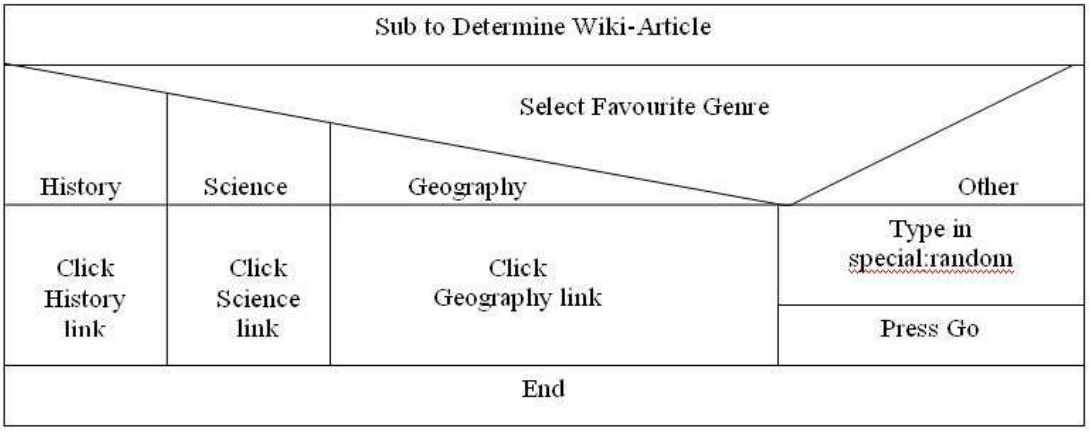

Figure 48: Example Nassi-Shneiderman diagram

4.4.5.1.11 Feature Summary

No tool provide the same level of code generation as VIDE toolset. BLU AGE from IBM is the closest one even though it is far from PIM. MID also provide some very limited behaviour modelling and code generation.

IBM (BLU AGE)	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.4.5.2 Complete and compile-ready code generation

This section should be read with the previous one in mind; we consider only tools that provide some form of dynamic modelling at PIM level and code generation.

4.4.5.2.1 IBM Rational Software Modeller

Complete and compile-ready code generation is available with BLU AGE.

4.4.5.2.2 Borland Together

Complete and compile-ready code generation is not available

4.4.5.2.3 Visual paradigm for UML

Complete and compile-ready code generation is not available

4.4.5.2.4 Telelogic Modeller

Complete and compile-ready code generation is not available

4.4.5.2.5 Topcased

Complete and compile-ready code generation is not available

4.4.5.2.6 Artisan Studio

Complete and compile-ready code generation is not available

4.4.5.2.7 openAmeos

Complete and compile-ready code generation is not available

4.4.5.2.8 MagicDraw

Complete and compile-ready code generation is not available

4.4.5.2.9 Select Solution for MDA

Complete and compile-ready code generation is not available

4.4.5.2.10 MID Innovator

Complete and compile-ready code generation is not available

4.4.5.2.11 Feature Summary

IBM (BLUAGE)	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	MagicDraw	Select	MID

Key:

No cover	
Partial	
Complete	

4.5 Comparison Matrix

	Evaluation Criteria	IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	open Ameos	Magic Draw	Select	MID
Pre-CIM	• Creating and storing descriptions of the domain (including descriptions of business requirements and operational procedures) .										
	• Selection of desired scrap text from the domain description to create a visual informal scrap model.										
	• Selection of scrap model elements for further analysis and creation of an informal analysis model										
	• Refinement of domain model elements into roles, activities or data objects.										
	• Selection of analysis model elements for use in creating an initial business process model										

Key:

No cover	
Partial	
Complete	

	Evaluation Criteria	IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	Magic Draw	Select	MID
CIM	• Allows the import of un-structured requirements information										
	• Allows the creation of Business Rules from both data structure and organisational structure model elements in addition to natural language										
	• Enhances business process modelling using extended BPMN • Creating such things as multi-media artifact information • Creating data elements, organisational data and business rules inside the process. (These can be used singly or combined.)										
	• Allows the creation of organisational and data structures using elements helpful to the non-IT person										
	• Exports information from the process model as XPDL (for a workflow engine.)										
	• Uses the export of a process model as a blueprint to allow creation of UML2 diagrams										

Key:

No cover	
Partial	
Complete	

	Evaluation Criteria	IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	Magic Draw	Select	MID
CIM TO PIM transition	• Automatic creation of class model from business process model (by use of heuristics for BPM-to-Class mapping).										
	• Automatic creation of activity chart from business process model (by use of heuristics for BPM-to-Activity mapping).										

Key:

No cover	
Partial	
Complete	

	Evaluation Criteria	IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	Magic Draw	Select	MID
PIM level	• Model execution at the PIM level										
	• Textual language and editor for UML action modelling										
	• Visual language and editor for UML action modelling										
	• User friendly action modelling										
	• OCL used as a high-level expression and query language										
	• OCL query expressions visualized										
	• PIM-level aspect-oriented modelling										
	• PIM-level quality defect detection										
	• PIM-level publication of web services - modelling and execution										
	• PIM-level consumption of web services - modelling and execution										
	• Platform independent means for GUI development										
	• Availability of a logical structure of GUI										

Key:

No cover	
Partial	
Complete	

	Evaluation Criteria	IBM	Borland	Visual Paradigm	Telelogic	Eclipse	Artisan	Open Ameos	Magic Draw	Select	MID
PIM/PSM level	• Code generation of behaviour from PIM (method/constructor bodies)										
	• Complete and compile-ready code generation (including web services)										

Key:

No cover	
Partial	
Complete	

4.6 Vertical evaluation conclusions

This chapter outlined the vertical evaluation of the VIDE toolset and shows that VIDE has 24 functional features which will help business users create data intensive applications, and which are missing from the major MDA toolchains.

Section 4.2 outlined the value added features of the VIDE components at the different MDA levels. Section 4.2.2 outlined the stakeholders and VIDE functionality at the pre-CIM level, before defining five value added features. This was repeated for the CIM level in Section 4.2.3 where 6 features were outlined and the CIM – PIM transition in Section 4.2.4 where 2 further features were defined. Section 4.2.5 discussed the PIM level and its stakeholders and identified 12 features which added value to the VIDE toolset and finally The PIM to PSM transition was discussed in Section 4.2.6 when a further two value added features were identified. Thus 24 features were selected in all. Section 4.3 then presented the ten toolchains which were selected to compare the VIDE toolchain against. These were selected on the basis that they covered the entire scope of the VIDE toolset and that they were well known in the MDA and business development environment. Information on each toolchain was made available including licensing arrangements to allow readers to compare the toolset against the available components named.

Section 4.4 details the presentation of the results of the investigation for each toolchain, feature by feature. These results are presented in a comparison matrix in Section 4.5. The evaluation of VIDE against the named tool chains shows that VIDE has 24 functional features that are mostly missing in all the major tool chains.

5 Conclusions

The evaluation work described in this deliverable intended to validate the measurable results assumed in the Description of Work and further requirements specified in the course of the project, consists of two main views.

One of them was called "vertical", as it is focused on comparing whole toolchains across the modelling layers and development process phase against the VIDE tool set. 27 essential features of VIDE were identified, classified into modelling layers:

- pre-CIM
- CIM
- CIM-to-PIM
- PIM
- PIM-to-PSM and code

and compared with 10 commercially available toolchains of similar area of application. The comparison has shown that the VIDE features identified are indeed unique – either completely, or at least in combination inside particular toolchains.

The other part of evaluation work was called "horizontal" as it was aimed at investigating selected features of VIDE inside particular layers of the framework. This part complements the vertical evaluation by checking if the VIDE unique features identified there actually bring an advantage. In other words, the workshops, competitions and study performed there have the purpose of verifying the expectations expressed in the project's description of work Measurable Results and the other intended features of the VIDE tooling. The possibilities for experimentation and comparison were limited by the fact that VIDE constitutes an proof of concept prototype implementation, which makes it problematic to compete with mature industrial tools in terms of usability. Nevertheless, most of the horizontal evaluation work was possible to be performed in the form of workshop using actual working VIDE functionality and comparing it against traditional solutions.

The results can be considered promising. They suggest a significant productivity gain resulting from the simplicity and seamless design of the VIDE PIM language. The visual syntax was found intuitive for novice users, while the visual expression solution substitutive to OCL textual coding allowed them to realise non trivial queries. The comparative analysis of the VIDE AO features confirm they offer at the PIM level analogous expressive power as the AO solutions of programming languages. The workshop on VIDE Defect Detection shows that the tool allows a significantly higher quality defect detection rate than a manual inspection. Also the non-IT user oriented VIDE solutions offered by the "pre-CIM" tooling have been found, in the course of workshop experimentation, much more approachable than traditional modelling solutions that would constitute a communication barrier for domain experts.

References

1. Iris Groher, Stefan Schulze: Generating aspect code from UML models, The 4th Aspect-Oriented Software Development Modeling With UML Workshop. 2003
2. Abeywickrama, D. and Ramakrishnan, S. 2008. Towards engineering models of aspectual pervasive software services. In *Proceedings of the 3rd ACM Workshop on Software Engineering For Pervasive Services* (Sorrento, Italy, July 06 - 06, 2008). SEPS '08. ACM, New York, NY, 3-8.
3. Whittle, J. and Jayaraman, P. 2008. MATA: A Tool for Aspect-Oriented Modeling Based on Graph Transformation. In *Models in Software Engineering: Workshops and Symposia At MoDELS 2007, Nashville, Tn, Usa, September 30 - October 5, 2007, Reports and Revised Selected Papers*, H. Giese, Ed. Lecture Notes In Computer Science, vol. 5002. Springer-Verlag, Berlin, Heidelberg, 16-27.
4. Fuentes, L., Gamez, N., and Sanchez, P. 2008. Aspect-Oriented Executable UML Models for Context-Aware Pervasive Applications. In *Proceedings of the 2008 5th international Workshop on Model-Based Methodologies For Pervasive and Embedded Software - Volume 00* (April 05 - 05, 2008). MOMPES. IEEE Computer Society, Washington, DC, 34-43.
5. Cottenier, T., van den Berg, A., Elrad, T. The Motoroal WEAVR: Model Weaving in a Large Industrial Context. Industry Track paper to be presented at AOSD'07, 2006
6. D. Stein, S. Hanenberg, and R. Unland. Expressing different conceptual models of join point selections in aspect-oriented design. In AOSD '06: Proceedings of the 5th international conference on Aspect-oriented software development, pages 15{26, New York, NY, USA, 2006. ACM.
7. Groher, I. and Voelter, M. 2007. XWeave: models and aspects in concert. In *Proceedings of the 10th international Workshop on Aspect-Oriented Modeling* (Vancouver, Canada, March 12 - 12, 2007). AOM '07, vol. 209. ACM, New York, NY, 35-40.
8. Cohen, J. (1988). Statistical Power Analysis for the Behavioral Sciences (2nd Edition): Lawrence Erlbaum.
9. ISO/IEC-9126-1. (2003). Software engineering: product quality. Part 1, Quality model (Ed. 1. ed.). Pretoria: International Organization for Standardization / International Electrotechnical Commission.
10. Ortega, M., Perez, M., & Rojas, T. (2003). Construction of a Systemic Quality Model for Evaluating a Software Product. Software Quality Journal, 11(3), 219-242.
11. Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User acceptance of information technology: Toward a unified view. MIS Quarterly, 27(3), 425-478.
12. VIDE-Annex. (2005). Annex I - "Description of Work" amendment (Revision 5): PJIIT.
13. Al-Khilidar, H, Baber, M.A., Berry, M., Cox, K., Keung, J., Kitchenham, B., Kurniawati, F., Liming, Z., Staples, M. and Zhang, H., 2007. Evaluating Guidelines for Reporting Empirical Software Engineering Studies. *Empirical Software Engineering*, 13 (1), 97-121.
14. Al-Kilidar, H., Aurum, A., Jeffery, R. and Parkin, P., 2005. valuation of Effects of Pair Work on Quality of Designs. *In: Software Engineering Conference, 29th March – 1st April 2005, Australia.*

15. Budgen, D., and Thomson, M., 2003. CASE Tool Evaluation: Experiences from an Empirical Study. Elsevier Science Inc, 67 (2), 55-75.
16. Lidia Fuentes and Pablo Sánchez. *"Designing and Weaving Aspect-Oriented Executable UML models"*. Journal of Object Technology - Special Issue on Aspect-Oriented Modelling.
17. Aspectj homepage, November 2008. http://www.eclipse.org/aspectj/.
18. E. Hilsdale, J. Hugunin, Advice Weaving in AspectJ, Mar 2004, AOSD04
19. JBoss AOP homepage, November 2008. http://www.jboss.org/jbossaop/.
20. Wim Vanderperren, Davy Suv´ee, Bruno De Fraine, and Viviane Jonckers. (2005): Aspect-Oriented Programming using JAsCo.
21. JAsCo homepage, November 2008. http://ssel.vub.ac.be/jasco/.
22. Johan Brichau, Wolfgang De Meuter, and Kris de Volder. Jumping aspects. In 14th European Conference on Object-Oriented Programming, 2000.
23. Klaus Ostermann, Mira Mezini, and Christoph Bockisch. Expressive Pointcuts for Increased Modularity. European Conference on Object-Oriented Programming (ECOOP'05), Springer LNCS, 2005
24. D. Stein, S. Hanenberg, and R. Unland. Expressing different conceptual models of join point selections in aspect-oriented design. In AOSD '06: Proceedings of the 5th international conference on Aspect-oriented software development, pages 15{26, New York, NY, USA, 2006. ACM.
25. LogicAJ homepage, November 2008. http://roots.iai.uni-bonn.de/research/logicaj/.
26. Günter Kniesel, Tobias Rho. Generic Aspect Languages - Needs, Options and Challenges. JFDLPA 2005. September 2005.
27. LogicAJ2 homepage, November 2008. http://roots.iai.uni-bonn.de/research/logicaj/logicaj2/

Appendix A: VIDE Defect Detector evaluation questionnaires

Briefing Questionnaire

Please answer the following questions. This will take you about 5 minutes. During the analysis of the data, the data will be anonymized – your name will be removed.

Subject-ID	**<the ID will be inserted by the evaluators>**	
Name:		Start Time:
ID:		End Time:

University Education

<B1>	**Education**	
<B1.1>	Name of study (e.g., “Angewandte Informatik”)	
<B1.2>	Major Subject (i.e., “Hauptfach/Vertiefung”):	
<B1.3>	Minor Subject (i.e., “Nebenfach/Wahlfach”): *(if more than one, please mention all)*	
<B1.4>	Which lectures regarding “Software Engineering” (e.g., “SE 1-3”, “GSE”, …) have you completed?	
<B1.5>	Number of terms (Fachsemester) completed (including the current one):	
<B1.6>	In how many practical courses (i.e., SE-oriented “Praktika”) have you participated?	

Practical Software Engineering Experience

<B2>	**Practical Software Engineering Experience**	**Yes**	**No**
<B2.1>	Have you ever written software system with more than 5 classes or 1000 lines of code?		
<B2.2>	Have you ever written software outside of university programs (e.g., private, commercial, OSS)?		
<B2.3>	Have you developed software in a large team (>4 persons) with distributed roles?		
<B2.4>	Have you developed software in a project with long duration (>6 months)?		

Experience with Programming & Object-oriented Languages

<B3>	Questions on Experience with Programming & Object-oriented Languages	
<B3.1>	How many years of computer programming experience do you have, if any?	
<B3.2>	How many different applications have you programmed?	
<B3.3>	How many different applications have you programmed in object-oriented languages?	
<B3.4>	How many years were you involved in maintaining & improving a software system?	

<B4>	What is your experience with …	High Experience						No Experience
<B4.1>	Object-oriented Languages APIs (such as java.util, java.io, java.net, etc.)	○	○	○	○	○	○	○
<B4.2>	Object-oriented Languages GUIs (such as AWT, Swing, SWT, etc.)	○	○	○	○	○	○	○
<B4.3>	Creating object-oriented programs from scratch	○	○	○	○	○	○	○
<B4.4>	Debugging large programs in object-oriented languages	○	○	○	○	○	○	○
<B4.5>	The eclipse IDE (as an user, not plugin-Developer)	○	○	○	○	○	○	○
<B4.6>	Other IDE such as Netbeans, Visual Studio, jBuilder, etc. (as an user, not plugin-Developer)	○	○	○	○	○	○	○

Experience with Refactoring & Code Smells

<B5>	General Questions	
<B5.0>	Have you heard of refactoring before?	
<B5.1>	How many years of experience do you have with refactoring?	
<B5.2>	How many different applications have you refactored? (all object-oriented languages)	
<B5.3>	How many different applications have you refactored in Java?	

<B6>	What is your practical experience with …	High Experience						No Experience
<B6.1>	Identifying code smells, anti-patterns, pitfalls, design flaws, etc.	○	○	○	○	○	○	○
<B6.2>	Applying Refactorings manually	○	○	○	○	○	○	○
<B6.3>	Applying Refactorings such as “Extract Method” build into an IDE (except the “rename” refactoring)	○	○	○	○	○	○	○
<B6.4>	Working with design patterns, design heuristics, design principles, etc.	○	○	○	○	○	○	○

Experience with Quality Assurance & Maintenance

<B7>	What is your practical experience with …	High Experience						No Experience
<B7.1>	Quality models (such as ISO 9126, FURPS, Dromey, Boehm, …)	○	○	○	○	○	○	○

<B7.2>	Testing a software system?	○	○	○	○	○	○	○
<B7.3>	Inspecting a software system regarding quality issues?	○	○	○	○	○	○	○
<B7.4>	Software measurement (Metrics)?	○	○	○	○	○	○	○
<B7.5>	Code checking tools such as PMD, checkstyle, etc.?	○	○	○	○	○	○	○

<B8>	**What is your practical experience with …**	**High Experience**						**No Experience**
<B8.1>	Maintaining a software system? (e.g., managing defects, applying changes, etc.)	○	○	○	○	○	○	○
<B8.2>	Porting a software system to another platform? (e.g., Java 1.2 to 5.0, Java to C#, etc.)	○	○	○	○	○	○	○
<B8.3>	Improving a software system regarding efficiency (time behaviour, resource behaviour)?	○	○	○	○	○	○	○
<B8.4>	Improving a software system regarding reliability? (i.e., “Zuverlässigkeit”)	○	○	○	○	○	○	○
<B8.5>	Improving a software system regarding usability?	○	○	○	○	○	○	○
<B8.6>	Improving a software system regarding functionality (suitability, interoperability, security)?	○	○	○	○	○	○	○

Learning Style

<B9>	**What is your most preferred learning style? (select one option)**	
<B9.1>	Reading textbooks (with exercises)	○
<B9.2>	Classroom lectures (with exercises)	○
<B9.3>	Group work (interaction with peers and teacher / including exercises)	○
<B9.4>	Web-based training modules (with computer interaction / including examples and exercises)	○
<B9.5>	Trial and error approach (e.g., program, debug, repeat)	○

Experience with Software Modelling & UML

<B10>	**What is your experience with …**	**High Experience**						**No Experience**
<B10.1>	The UML / UML diagrams	○	○	○	○	○	○	○
<B10.2>	Modeling of software systems (not necessarily with UML)	○	○	○	○	○	○	○
<B10.3>	Creating software using UML from scratch	○	○	○	○	○	○	○
<B10.4>	Debugging large software models	○	○	○	○	○	○	○
<B10.5>	Using the Topcased modeling environment	○	○	○	○	○	○	○
<B10.6>	Other modeling environments such as Objecteering, Omondo, etc.	○	○	○	○	○	○	○

<B11>	**Questions on Experience with Modeling & UML**	
<B11.1>	How many years of modeling experience do you have, if any?	
<B11.2>	How many different applications have you modeled?	
<B11.3>	How many different applications have you modeled using UML?	
<B11.4>	How many years were you involved in maintaining & improving a software model?	

Refactoring Protocol

Please answer the following questions. During the analysis of the data, the data will be anonymized – your name will be removed.

Subject-ID	**<the ID will be inserted by the evaluators>**	
Name:		Start Time:
ID		End Time:

Class	**Element**	**Smell - Refactoring / Decisions (**
Test.java	*getString()*	*Unc. Name – **changed** to getSurename()*
	calculateOffset()	*Long Method – **ignored** (necessary due to complex algorithm)*

Class	Element	Smell - Refactoring / Decisions (

Post-Test Questionnaire

Please answer the following questions. This will take you about 5 minutes. During the analysis of the data, the data will be anonymized – your Name will be removed.

Subject-ID	<the ID will be inserted by the evaluators>	
Name:		Start Time:

ID		End Time:

Quality Improvement (Subjective)

The following questions are targeted towards the perceived effects of the manual approach or VIDE-DD on the software quality (Based on ISO 9126)

Maintainability

The effort needed to make specified modifications.

VIDE-DD improved the following aspects of Maintainability	**Agree**				**Disagree**		
Analyzability: *Attributes of software that bear on the effort needed for diagnosis of deficiencies or causes of failures, or for identification of parts to be modified.* (ISO 9126: 1991, A.2.5.1)	○	○	○	○	○	○	○
Changeability: *Attributes of software that bear on the effort needed for modification, fault removal or for environmental change.* (ISO 9126: 1991, A.2.5.2)	○	○	○	○	○	○	○
Stability: *Attributes of software that bear on the risk of unexpected effect of modifications.* (ISO 9126: 1991, A.2.5.3)	○	○	○	○	○	○	○
Testability: *Attributes of software that bear on the effort needed for validating the modified software.* (ISO 9126: 1991, A.2.5.4)	○	○	○	○	○	○	○
Understandability (Code), Readability: *Attributes of software to be understood by the developer, tester, or maintainer.* (Boehm)	○	○	○	○	○	○	○

Portability

The ability of software to be transferred from one environment to another.

VIDE-DD improved the following aspects of Portability	**Agree**				**Disagree**		
Adaptability: *Attributes of software that bear on the opportunity for its adaptation to different specified environments without applying other actions or means than those provided for this purpose for the software considered.* (ISO 9126: 1991, A.2.6.1)	○	○	○	○	○	○	○
Installability: *Attributes of software that bear on the effort needed to install the software in a specified environment.* (ISO 9126: 1991, A.2.6.2)	○	○	○	○	○	○	○
Conformance: *Attributes of software that make the software adhere to standards or conventions relating to portability.* (ISO 9126: 1991, A.2.6.3)	○	○	○	○	○	○	○
Replaceability: *Attributes of software that bear on the opportunity and effort of using it in the place of specified other software in the environment of that software.* (ISO 9126: 1991, A.2.6.4)	○	○	○	○	○	○	○

Efficiency

The relationship between the level of performance of the software and the amount of resources used, under stated conditions.

VIDE-DD improved the following aspects of Efficiency	**Agree**				**Disagree**		
Time behaviour: *Attributes of software that bear on response and processing times and on throughput rates in performing its function.* (ISO 9126: 1991, A.2.4.1)	○	○	○	○	○	○	○
Resource behaviour: *Attributes of software that bear on the amount of resources used and the duration of such use in performing its function.* (ISO 9126: 1991, A.2.4.2)	○	○	○	○	○	○	○

Reliability

The capability of software to maintain its level of performance under stated conditions for a stated period of time.

VIDE-DD improved the following aspects of Reliability	**Agree**				**Disagree**		
Maturity: *Attributes of software that bear on the frequency of failure by faults in the software.* (ISO 9126: 1991, A.2.2.1)	○	○	○	○	○	○	○
Fault tolerance: *Attributes of software that bear on its ability to maintain a specified level of performance in cases of software faults or of infringement of its specified interface.* (ISO 9126: 1991, A.2.2.2)	○	○	○	○	○	○	○
Recoverability: *Attributes of software that bear on the capability to re-establish its level of performance and recover the data directly affected in case of a failure and on the time and effort needed for it.* (ISO 9126: 1991, A.2.2.3)	○	○	○	○	○	○	○

Functionality

The existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.

VIDE-DD improved the following aspects of Functionality	**Agree**				**Disagree**		
Suitability: *Attribute of software that bears on the presence and appropriateness of a set of functions for specified tasks.* (ISO 9126: 1991, A.2.1.1)	○	○	○	○	○	○	○
Accuracy: *Attributes of software that bear on the provision of right or agreed results or effects.* (ISO 9126: 1991, A.2.1.2)	○	○	○	○	○	○	○
Interoperability: *Attributes of software that bear on its ability to interact with specified systems.* (ISO 9126: 1991, A.2.1.3)	○	○	○	○	○	○	○
Compliance: *Attributes of software that make the software adhere to application related standards or conventions or regulations in laws and similar prescriptions.* (ISO 9126: 1991, A.2.1.4)	○	○	○	○	○	○	○
Security: *Attributes of software that bear on its ability to prevent unauthorized access, whether accidental or deliberate, to programs and data.* (ISO	○	○	○	○	○	○	○

9126: 1991, A.2.1.5)							

Usability

The effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.

VIDE-DD improved the following aspects of Usability	**Agree**				**Disagree**		
Understandability (System): *Attributes of software that bear on **the users' effort** for recognizing the logical concept and its applicability.* (ISO 9126: 1991, A.2.3.1)	○	○	○	○	○	○	○
Learnability: *Attributes of software that bear on **the users' effort** for learning its application (for example, operation control, input, output).* (ISO 9126: 1991, A.2.3.2)	○	○	○	○	○	○	○
Operability: *Attributes of software that bear on **the users' effort** for operation and operation control.* (ISO 9126: 1991, A.2.3.3)	○	○	○	○	○	○	○

Debriefing Questionnaire

Please answer the following questions. This will take you about 5 minutes. During the analysis of the data, the data will be anonymized – your Name will be removed.

Subject-ID	**<the ID will be inserted by the evaluators>**	
Name:		Start Time:
ID		End Time:

<J1>		**Agree**				**Disagree**		
<J1.1>	I found the presentation of quality defects helpful	○	○	○	○	○	○	○
<J1.2>	I found the presentation of refactorings helpful	○	○	○	○	○	○	○
<J1.3>	The complexity of both runs were comparable	○	○	○	○	○	○	○
<J1.4>	The system in Run 1 (SalesScenario System) was more complex	○	○	○	○	○	○	○
<J1.5>	The system in Run 2 (BelAMI System) was more complex	○	○	○	○	○	○	○
<J1.6>	The system in Run 1 (SalesScenario System) was harder to understand	○	○	○	○	○	○	○
<J1.7>	The system in Run 2 (BelAMI System) was harder to understand	○	○	○	○	○	○	○
<J1.8>	I have experience with the domain of system 1 (i.e., project acquisition / CRM)	○	○	○	○	○	○	○
<J1.9>	I have experience with the domain of system 2 (i.e., diagnostic systems / AMI)	○	○	○	○	○	○	○

<J1.10>	The time was sufficient to understand the system in Run 1 (SalesScenario System)	○	○	○	○	○	○	○
<J1.11>	The time was sufficient to understand the system in Run 2 (BelAMI)	○	○	○	○	○	○	○

Motivation

<M>	**Motivation**	**Agree**				**Disagree**		
<M1.1>	I was interested / motivated to perform in the experiment	○	○	○	○	○	○	○
<M1.2>	The topic was too new for me to comprehend it	○	○	○	○	○	○	○
<M1.3>	I would like to know more about refactoring and code smells	○	○	○	○	○	○	○
<M1.4>	The experiment kept me from doing more important work in the lab (i.e., “GSE Praktikum”)	○	○	○	○	○	○	○

I would like to make the following comment(s) / improvement suggestion(s) (can be in German)

I had a problem with … <please explain (can be in German)>:

Evaluation of the Use and Acceptance of VIDE-DD

The following questions are based on the UTAUT (Unified Theory of Acceptance and Use of Technology).

Performance expectancy	**Agree**						**Disagree**
I would find the system useful in my work.	○	○	○	○	○	○	○
Using the system enables me to accomplish tasks more quickly.	○	○	○	○	○	○	○
Using the system increases my productivity.	○	○	○	○	○	○	○
If I use the system, I will increase my chances of getting a raise. (e.g., by being faster)	○	○	○	○	○	○	○

Effort expectancy	**Agree**						**Disagree**
My interaction with the system would be clear and understandable.	○	○	○	○	○	○	○
It would be easy for me to become skillful at using the system.	○	○	○	○	○	○	○
I would find the system easy to use.	○	○	○	○	○	○	○
Learning to operate the system is easy for me.	○	○	○	○	○	○	○

Attitude toward using technology	**Agree**						**Disagree**
Using the system is a good idea.	○	○	○	○	○	○	○
The system makes work more interesting.	○	○	○	○	○	○	○
Working with the system is fun.	○	○	○	○	○	○	○
I like working with the system.	○	○	○	○	○	○	○

Social influence	**Agree**						**Disagree**
People who influence my behaviour think that I should use the system.	○	○	○	○	○	○	○
People who are important to me think that I should use the system.	○	○	○	○	○	○	○
The senior management has been helpful in the use of the system.	○	○	○	○	○	○	○
In general, the organization has supported the use of the system.	○	○	○	○	○	○	○

Facilitating conditions	**Agree**						**Disagree**
I have the resources necessary to use the system.	○	○	○	○	○	○	○
I have the knowledge necessary to use the system.	○	○	○	○	○	○	○
The system is not compatible with other systems I use.	○	○	○	○	○	○	○
A specific person (or group) is available for assistance with system difficulties.	○	○	○	○	○	○	○

Self-efficacy I could complete a job or task using the system…	**Agree**				**Disagree**		
… if there was no one around to tell me what to do as I go.	○	○	○	○	○	○	○
… if I could call someone for help if I got stuck.	○	○	○	○	○	○	○
… if I had a lot of time to complete the job for which the software was provided.	○	○	○	○	○	○	○
… if I had just the built-in help facility for assistance.	○	○	○	○	○	○	○

Anxiety	**Agree**				**Disagree**		
I feel apprehensive about using the system.	○	○	○	○	○	○	○
It scares me to think that I could lose a lot of information using the system by hitting the wrong key.	○	○	○	○	○	○	○
I hesitate to use the system for fear of making mistakes I cannot correct.	○	○	○	○	○	○	○
The system is somewhat intimidating to me.	○	○	○	○	○	○	○

Behavioural intention to use the system	**Agree**				**Disagree**		
I intend to use the system in the next 6 months.	○	○	○	○	○	○	○
I predict I would use the system in the next 6 months.	○	○	○	○	○	○	○
I plan to use the system in the next 6 months.	○	○	○	○	○	○	○