Method Evaluation in Practice: UML/RUP & Inspired Method

Graham McLeod
mcleod@iafrica.com
+27 82 578 1834
Inspired/ University of C.T.

Dirk Roeleveld
roelevel@iafrica.com
+27 82 862 4708
University of Cape Town

Abstract

A study by postgraduate students at the University of Cape Town compared the UML and Rational Unified Process (RUP) combination to the Inspired Method with respect to complexity, quality and efficacy using the Castellani and Method Points; Krogstie and colleagues; and NIMSAD approaches respectively. This paper summarises the findings, learnings and areas for future research from the earlier report. The results indicate that the UML is lacking in the area of business analysis, especially strategic alignment of information systems with business goals and business process modeling. The Inspired method offers improved business analysis support and a wider planning perspective, but does not provide as much detail in expressing software design as does the UML/RUP. The Inspired method provides a more economical notation, but is not well supported in third party tools. Observations are made regarding challenges of applying the methods evaluation approaches and areas for future research are suggested.

1. Introduction

Organisations spend very large sums on the development and integration of information systems. It is widely recognised that successful delivery of these systems requires use of an effective method. Choosing an effective method, and one that is appropriate to the industry, organization, available skills and technology employed is no easy matter, however. In recent years, a variety of approaches have been proposed to assist in evaluation of methods, many of these published in the proceedings of the Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD) workshops and the proceedings of the Conference on Advanced Information Systems Engineering (CAiSE). The literature is less abundant with respect to the application of the proposed evaluation methods.

Inspired is a company that has developed its own approach for systems delivery, the Inspired Method (IM). Inspired encouraged a research team from the University of Cape Town to objectively critique its method (IM) [McLeod, 2001; Inspired, 2001], and the Rational Unified Process (RUP) [Rational Software Corporation, 1999], which makes use of the Unified Modeling Language (UML) [Object Management Group, 2001]. The team consisted of three honours students with a reasonable exposure to both methods. The research was not intended to favour either method, but to rather find strengths and weaknesses of each. Another goal was to assess the usefulness and practicality of the evaluation techniques, applied by non-authors. The results [Yates, Roeleveld, Goosen, 2001] would provide useful information for method engineers and practitioners. Opinions were also sought regarding the viability of methods evaluation techniques. This paper is a summary of the research results. Notable omissions include the full literature review, and the method descriptions.

2. Research Approach

The study began by seeking suitable criteria and evaluation techniques with which to measure each method. From a study of the literature, especially prior EMMSAD Proceedings, expert interviews and a semi-formal survey, a list of ranked criteria for method evaluation was compiled (Appendix 1). The researchers attempted to gauge respondents' views as to how well respective methods met these criteria. The survey was frustrated by a low response rate, so an expert evaluation technique was substituted. Survey results did confirm the expert evaluation conclusions but were not of sufficient power to be reliable. The authors consider the criteria from the survey to be sound, and have included them in the appendix to assist others conducting similar research.

Three criteria were used for method evalution:
- Overall method **effectiveness** (how well can a method potentially solve a business problem related to implementation of an information system)
- **Modeling capability** of the method
- **Complexity** of the method - ease of adoption and use for typical practitioners

Published evaluation methods chosen to assess the above are introduced below.

2.1 Method Effectiveness – NIMSAD Method Evaluation Framework

The Normative Information Model-based, Systems Analysis and Design (NIMSAD) framework [Jayaratna, 1994] rates a method's ability to formulate a problem, design a solution, and implement the design. Since a systems analysis and design method is a structured technique for solving a particular kind of problem, NIMSAD can be used to determine the effectiveness of such methods to solve business problems. The framework focuses on the context for applying the method, the method user, and the method itself (problem-solving process).

2.2 Deliverable Quality - Krogstie et al

The Krogstie, Sindle and Lindland quality assessment framework [Krogstie, 2000] was designed to evaluate the potential of modeling techniques to support the creation of high quality models. It recognises that models are created as part of a dialogue of participants, whose knowledge improves with the development of the models. Quality in the framework examines correspondence between statements belonging to several sets, including modeling language extension, the problem domain, the resulting externalised model, the relevant explicit knowledge of the audience, the social audience interpretation of the externalised model, and the technical audience interpretation via modeling tools. We used 3 of 5 facets of the model to help assess modeling capability.

2.3 Ease of Adoption and Use

Various authors [Castellani, 1998a, Peleg & Dori, 2000, Krogstie, 2000, McLeod, 1997] conclude that methods that are complex and large (many concepts and different model types) are more difficult to learn and apply. Where symbols in a modeling language are imprecisely defined, or overloaded (used for multiple purposes), ambiguities can arise. Where there are multiple equivalent constructs, there is redundancy, leading to wasted learning effort and possible loss of communication effectiveness. We found two techniques which provided good insight into these problems for the methods evaluated.

Castellani - [Castellani, 1998a] has used Charts of Concepts to derive measures of size and complexity of UML both in and between models. This approach is applied to the IM for comparison.

Method Points - Is a methodology, akin to the Function Point count, providing a quantitative measure of complexity by counting the components of the generic structure of a method's deliverables [McLeod, 1997].

The techniques introduced above were applied to both methods. Where available, we used the authors' own analysis for the RUP/UML. Original analysis was performed for the IM.

3. Analysis of Effectiveness Using NIMSAD Framework

NIMSAD examines the methods from the perspective of three major categories: the problem situation (scoping and framing the problem), the method user (how practitioners perceive the problem, and their ability to utilize the method) and the actual process of solving the problem.

We will discuss the RUP/UML and the IM under each of the relevant headings.

3.1 The Problem Situation

The first step of the NIMSAD framework is to consider how the method examines the environment in which the problem occurs.

RUP   Use Case Models are the primary instruments to analyse the problem environment. There is no clear ability to represent the organisation in order to pinpoint problem owners and identify reasons for problems in UML. RUP does suggest optional business case. Clients' requirements are taken at face value.

        The RUP is best suited for well-structured problems and does not cover coping with 'political' problems of clients. Most of the management advice is for controlling risk within the project team, so change control (changing software) is emphasised over change management (helping system users to cope with the new software).

IM   Attempts to accurately conceptualise the problem situation by modeling the real world aspects, and not just those pertaining to a proposed system. IM includes stakeholder models and high-level business process models for this purpose. Method users are encouraged to use their expertise to better co-ordinate information systems with the overall organisational purpose. IM considers the net change from status quo via architecture "delta" models. Architectural models may, however, become quickly outdated and may require a substantial involvement from top management.

3.2 The Method User

Next, NIMSAD considers the perspective of the 'method user'. This person is the intended problem solver (typically a consultant or system analyst) who is using the method to solve a problem.

3.2.1 Before a method can be successfully used, the method *user* needs to have an understanding of the relevant concepts, and their use, as designed by the method *creator*.

RUP   and the UML were created by more than one author and are therefore an amalgam of different mindsets and ways of thinking. Such unification may bring a more widely applicable approach but poses additional learning difficulties for new method users. The UML 1.3 specification is over a thousand pages long and begins by explaining the language in terms of a meta-metamodel. This can create problems, as meta concepts are usually only understood after prior instance modeling experience. The UML is also a complex method with over 233 concepts for the new user to grasp [Castellani, 1998].

IM   Prefers the use of smaller teams of highly skilled individuals. They will have to understand the 153 discrete concepts [see later analysis] to use the method efficiently. IM is easier to learn (when compared with the UML) and has often been used to introduce students to the concepts of object modeling at the University of Cape Town. IM contains only three different types of diagrams derived from UML diagrams plus extensions.

        IM explains the rationale behind tasks that need to be performed, which empowers the method user to achieve the goals behind the tasks. Some Inspired approaches are different from those used in industry, and require the method user to commit more fully to object-oriented modeling and to have experience in these areas.

3.2.2. There needs to be an understanding between the method *user* and the *client* whose problem is to be solved.

RUP   The rigorous specification of the UML facilitates easy communication between users of the method. Various authors have, however, highlighted the inability of UML to fully capture requirements (especially non-system requirements) as well as inconsistent semantics in definition. Most of the inconsistencies are addressed by ongoing refinement of the UML. The ability to use more detailed UML models to communicate with clients is questionable. If the client is not familiar with object-oriented concepts then expecting them to understand sequence, collaboration and state diagrams may be a tall order. RUP mentions prototyping, but there is no modeling guidance or linkage within UML. The UML is, however, the closest the IT has as an industry standard for such communication.

IM   Rich business process models are used in joint modeling sessions with users. These models evolve in fidelity and detail as analysis and design progresses. IM makes use of prototypes to show functionality and to demonstrate ideas. Building prototypes may be an expensive route, but has definite advantages such as allowing clients to explore requirements and design options. The IM also recommends a joint application development (JAD) approach so that all stakeholders are able to reach consensus on important decisions, take ownership, and decrease delivery times. Inspired process models are less well-known and therefor cannot be communicated as easily to other teams not trained in the method.

        The complexity of each method is a fundamental factor in the learning process. Other analysis of the methods' complexity is also included in the paper under the heading of Complexity.

3.3 The Problem Solving Process

The third category of NIMSAD, the problem solving process, consists of a further eight stages, which are listed below. Stages one to five are classified as problem formulation tasks, while stages six and seven belong to solution design. The final, eighth, stage is the implementation of the design. Each is discussed below:

3.3.1 Problem Formulation

Stage 1: Understanding the Situation of Concern

RUP   The RUP recommends using business use cases for scoping purposes which have the advantage of being easy to use, and easy for clients to understand. The RUP does not give clear guidelines on how use cases should be analysed, to determine what is in scope or not. Business Analysis also seems to be an elective according to RUP: 'Many projects may choose to skip business modeling'. The consequences of neglecting such analysis tasks can be significant, as research shows that errors caused early on are the most costly to rectify. According to DeMarco [1979], 56% of errors arise in the requirements phase and account for 82% of costs to correct. RUP provides little discussion of analysing existing situations.

IM   To identify sections of the business that need to be addressed, IM uses a stakeholder model that depicts relevant parties interacting with the system. Boundary symbols show what is inside or outside the situation of concern. The IM advises a 'big picture' approach to understand the project priorities, client's motives and to cope with a changing organisational environment. Jayaratna [1994] notes that it is only by examining an information system in context of the organisation that you can improve its effectiveness.

Stage 2: Performing the Diagnosis

RUP   When considering user specifications, the RUP does not attempt to challenge any requirements, which are essentially entirely derived from clients. There is no real way to analyse whether automation of functionality would provide any form of cost benefits to the business. The UML does not attempt to benchmark business processes that it will affect in terms of costs, volumes, or process times.

IM   The IM advises that analysis is performed on the current situation to prevent the analyst from considering only the first solution that occurs to him/her. Business Object and Business Process Models are used for this purpose to represent the *things* and *activities* respectively.

    The Inspired Business Process Model records for each process whether it is manual, computer supported, or a fully automated process. In addition minimum, average, and maximum process times can be recorded. This allows statistical process optimisation to be performed, or at least provides a benchmark against which to compare the proposed systems. These performance statistics can be transformed into cost benefit information using gathered data or value chain approaches.

Stage 3: Defining the Prognosis Outline

Once the AS-IS models have been clarified, the intended problem solvers should determine what the 'desired situation' is. It is important to question requirements to check that they are sound and not based on unrealistic expectations. Often there may be a superior way of achieving the same goals.

RUP   In the RUP framework, the client's expectations are gathered in the requirements stage in the Inception phase. RUP makes a base assumption that an information system is the appropriate type of solution and does not really attempt to explore alternative approaches.

IM   encourages method users to identify unrealistic expectations and to find simpler, more efficient ways to solve business problems, especially in ways that contribute to the enterprise architecture goals.

Stage 4: Defining Problems

RUP   The process of identifying what problems are holding the company back from achieving its 'desired situation' may be helpful in designing the solution. This process is usually carried out using some form of gap analysis. The RUP is largely incapable of such an analysis because it does not collect any information on process effectiveness or on resource inefficiencies. The RUP does not offer any meaningful way to analyse bottlenecks in the business. Fishbone models are suggested as a way to diagnose quality problems.

IM   Having information available (such as the times needed to complete processes, the associated costs, and volumes) makes problem areas, such as bottlenecks, easier to pinpoint. In cases of ill-structured problems (where the problem is actually the formulation of requirements) method users can use the Inspired models to examine how current processes diverge from strategic goals. Competing solution scenarios can be compared.

    Once system goals have been identified, IM recommends including associated performance measures into the system to encourage a self-optimising process.

Stage 5: Deriving Notional Systems

A method needs to be able to create a range of hypothetical systems that may each potentially solve the business problem, but from different perspectives.

RUP   This process of devising systems that attempt to satisfy the agreed requirements is catered for by the *Analysis and Design* step, which is prominent mostly in the Elaboration phase. The RUP emphasises a robust design which is easily updateable should the requirements change. Notional systems are depicted in the logical system designs.

IM   To clarify an understanding and engage clients in designing the system, IM users would use prototypes to illustrate screens, reports, and the overall flow. Prototypes provide the benefit of giving system users a feel for the system without the full investment. Only different types of processes should be prototyped.

    The Inspired process models are at this stage developed further to include events. Changes in state of domain objects are recorded. Activities are decomposed until each results in state changes of no more than one domain object type (class). This implies well-defined granularity levels. The Inspired Process supports reengineering of business processes to achieve outputs more effectively, efficiently, or at higher quality. Method users are encouraged to treat any business process reengineering with sensitivity and caution.

3.3.2 Solution Design

Stage 6: Performing Conceptual / Logical Design

The conceptual/logical design should show the roles of system users and the way in which the proposed system will link to the 'situation of concern'.

RUP The UML has many conceptual design models that can be used for this purpose. Examples include the static class diagram or the dynamic activity and statechart Diagrams.

  The RUP recommends splitting the software system into subsystems and grouping classes into *packages.* Decoupling modules is good practice because it increases system maintainability, and facilitates component reuse.

  The RUP encourages design of interfaces (Component Diagram) at this stage, and modeling of how Classes collaborate to realise Use Cases (Collaboration Diagrams). The RUP has a good foundation for rigorous architectural analysis because the method user creates models from many different perspectives.

IM Mapping of behaviour to classes occurs in the system design phase. The system is partitioned into classes which represent business entities, those responsible for carrying out system activities (business logic), and user or other interfaces to the system. This is in accordance with the three-layered *Model, View, Controller* framework.

  Encapsulating business logic into its own layer promotes future integration with other systems. The user interface layer, in contrast, does not contain any system functionality thus allowing many interfaces to the same underlying business functionality.

  Inspired also tries to speed up delivery by reusing code from class libraries and by solving generic type problems with the use of patterns. Business rules are captured in the Process Diagrams.

  IM does not perform any state transition analysis for each object, , but effects of activities on domain objects are recorded on event models derived from business process models.

Stage 7: Performing Physical Design

The logical design models created are later linked to physical entities.

UML provides Component and Deployment Diagrams for this purpose. The UML is able to represent linking with source files, binaries, and executables clearly. The RUP also encourages linking system components to development teams to improve development productivity.

IM optionally groups functionality on design level activity models into components via bounding boxes. Platform mappings are supported via similar bounding boxes. Normally, it advises that required functions are mapped directly to classes using the layered architecture mapping, which is well described. Domain objects are, in terms of the method, ideally implemented as persistent objects either in an object-oriented database or by making use of relational wrappers.

  A Technical Environment Model is provided by IM to represent the interconnection of modules, platforms, software, and geographical distribution of the systems. The objects are then mapped to layers in the Technical Model.
  3.3.3 Design Implementation

3.3.3 Design Implementation

Stage 8: Implementing the Designs

RUP The process of creating an implementation strategy is simplified by the RUP, which links the UML packages (groups of classes) directly to implementation subsystems. In order for a system to be considered complete the RUP makes sure to check that all client's requirements have been fulfilled. All components have to be correctly tested, and all errors have to be fixed prior to deployment.

  The RUP advocates the use of automated system tests. This has the advantage of performing multiple tests in short spaces of time, and the disadvantage of potentially overlooking errors not covered by the test routine.

  Good recommendations are made by the RUP in terms of coping with changes in the software and documents using change control. However, the impact of new software on people, in terms of change management, is virtually ignored.

IM In the implementation stages, the IM advises testing of functionality, and aims to provide quality assurance based on its project management foundation.

Various CASE tools or a rapid application development environment are recommended to translate the business models into a working system. One can see that the IM is not as concerned with the detailed implementation as it is with sound business analysis and architectural design.

4. Deliverable Quality - Krogstie et al.

The model, through checking correspondence between modeling domain, participant knowledge, social actor interpretation, language extension, model externalization and technical actor interpretation, derives opinions for different quality types, including:

- Physical quality – knowledge of stakeholders has been externalised and captured in the model; stakeholders are able to understand and internalise the model
- Empirical quality – model is robust enough to be understood and updated by various users with low error rates
- Syntactic quality – syntax valid for the model type and consistent with modeling language extensions
- Semantic quality – model is both valid (statements are correct and relevant to the problem), and complete (statements encompass all required domain knowledge)
- Perceived Semantic quality – user's current explicit knowledge agrees with their interpretation of the model
- Pragmatic quality – model is the same as how the audience actually understands it and participants agree on interpretation

The framework addresses two perspectives of a modeling language:
- The constructs of the language (concepts and meta-model)
- How the constructs are visually represented (notation).

The application of the framework requires analysis of the two perspectives, using five main quality groups:
- Domain Appropriateness - All statements in the domain should be expressible, and the language should not allow for statements outside of the domain to be expressed. It can, therefore, be said that different languages are appropriate for different problems. The implications for the language under consideration are that:
  - the phenomena must be general rather than specialised
  - the phenomena must be composable, meaning that statements can be grouped in a related way
  - the language must be flexible in its level of precision
  - Domain appropriateness is primarily a means to achieve physical quality.
- Participant language Knowledge Appropriateness - All statements made in the modeling language are explicit knowledge of the participants; therefore the conceptual basis must correspond with the way in which the participant perceives the problem. Participant language knowledge appropriateness is primarily a means to achieve physical and pragmatic quality
- Knowledge Externalisability Appropriateness - There should be no statements in the participants' knowledge that cannot be expressed in the language. Knowledge externalisability appropriateness is primarily a means to achieve physical quality
- Comprehensibility Appropriateness - Participants in the modeling effort must understand all possible statements of the language. This means that:
  - the language phenomena should be easily distinguishable
  - the number of the phenomena should be reasonable
  - the use of phenomena should be uniform
  - the language must be flexible in the level of detail
  - the language must allow for separation into areas of concern
  - the language must have expressive economy (the most frequent and important statements are brief)
  Comprehensibility appropriateness is primarily a means to achieve empirical and pragmatic quality.
- Technical Actor Interpretation Appropriateness - This relates the language to the technical audience interpretation, which should lend itself to automatic reasoning through formality and executability. The power of formal semantics lies in three aspects:
  - The process of making a more formal specification may reveal errors and ambiguities at an early stage in the development process
  - Formal and even automated proofs may be available
  - The remaining (or unprovable) rules may be translated into executable constraints in some imperative language
  The different aspects of technical actor interpretation appropriateness are a means for achieving syntactic, semantic and pragmatic quality.

We used the Domain Appropriateness, Comprehensibility Appropriateness and Technical Actor Interpretation Appropriateness facets of the model, as these are context independent. The remaining two facets deal closely with "participants" and their analysis would require the context of a project or development initiative. Analysis

of the specific quality types as specified by the framework was also omitted, as this is dependent on the tool used to render the language. The evaluation of tools falls outside the scope of this paper.

Krogstie [2000] has already used his quality framework to analyse the UML in terms of the above-mentioned categories. A similar exercise upon the IM deliverables yielded useful results regarding potential deliverable quality.

In terms of **Domain Appropriateness**, there are areas in which both the UML and Inspired fall short of being able to model all statements possible in a given domain. Neither language supports the modeling of organisational or group structures, and both allow for the use of mechanisms appropriate in design stages during the analysis stage. However, the IM does so with appropriate instruction, to allow for ease of generation of models for the consecutive stages in the development process.

In terms of **Comprehensibility Appropriateness**, it appears that there are some symbol duplications and differentiation discrepancies with both methods. This is however, less pronounced in the IM as it was developed with the aim of addressing this very problem in the UML, as identified by [Castellani, 1998b]. IM also allows for more flexibility in the level of detail shown in its models.

Technical Actor Interpretation Appropriateness deals with the technical proficiency of the language and hence the degree to which it lends itself to automatic and logical reasoning. There are identified instances in the UML where mechanisms and models do not follow such reasoning. The IM allows for some degree of automated checking and may be considered less technical than the UML.

Both methods fall short of perfect quality as Krogstie predicted all methods would. However, in most respects, the IM has greater potential to produce models of a high quality standard.

5. Determining Complexity

The ability to learn a method's concepts quickly and to be able to competently apply the correct mental processes to these concepts depends heavily on the **complexity** of the method. A method may be useful but impractical because performing the required tasks is impossible given the current skills of the available people. High complexity will also have a detrimental affect on the ability of the models to serve as an efficient communication medium. In the following section Castellani's [1998a] proposed size and complexity metrics are applied to both the UML and the IM.

5.1 Chart of Concepts

Castellani has previously applied the metrics to the UML 1.1 [1998a, 1998b] and his findings serve as a basis of comparison with the IM. Using his approach, the discrete concepts and their inter-relationships of the IM were mapped. The charts of concepts for the IM are included in Appendix 2.

Basic Concepts	UML	Inspired
Concepts relative to objects	24	24
Concepts relative to classes	8	8
Concepts relative to auxiliary elements	12	12
Concepts relative to extension mechanisms	8	8
Concepts relative to data	32	32
Total number of basic concepts	*84*	*84*

Diagrams		
Use Case diagram	7	0
Class Diagram	42	44
Object Diagram	7	0
Collaboration Diagram	9	0
Sequence Diagram	23	0
Statechart Diagram	33	0
Activity Diagram	6	0

Component Diagram	11	0
Deployment Diagram	11	0
Process Diagram	0	25
Total number of concepts of charts of diagrams	*149*	*69*
Number of Concepts	**233**	**153**

Table 1 - Total number of discrete concepts

It can be concluded that the UML is considerably larger than IM (by a factor of 52%). It should be noted that, because both methods share the same object oriented concepts, the size difference is due to Inspired's more concise diagramming. The number of concepts used in the UML diagrams (149) is more than double the number used in diagrams of the IM (69).

Complexity

Castellani proposes the measure of complexity of a model with a number of metrics. These are detailed in the table below with the results for both the UML and the IM.

Metric	UML	IM
Overlapping - The number of concepts that are common between charts of concepts.	49	7
Binding - The number of charts of concepts used to define concepts of other charts. Complexity increases with this metric as the ability to understand concepts becomes increasingly dependent on having understood other charts of concepts.	57	6
Definition Dependencies - A ratio of the number of definition dependencies to the number of concepts. Provides an indication of how inter-related concepts are.	313:233 1.34	148:153 0.97
Average Number of Concept Predecessors - Measures the average number of concept predecessors per concept. Therefore, on average, how many concepts need to be understood as a prerequisite to understanding new ones.	1.38	1.24
Average Number of Successors - Measures the average number of concept successors per concept. Therefore, the average number of concepts dependent on understanding a concept.	1.36	2.04
Number of Root Concepts - A factor of complexity as it is the number of concepts from which all concepts are defined.	14	26
Number of Terminal Concepts - A factor of complexity as it is the number of concepts from which no other concepts stem.	129	74
Cyclomatic Complexity - Originally used by Mccabe for measuring the complexity of software modules. Castellani uses the same metric in his chart of concepts. It is a representation of the number of independent dependency paths between root and terminal nodes.	82	5

Table 2 - Complexity metrics for the UML and the IM

From the above it can be concluded that:
- IM has fewer concept bindings and overlaps. This is because Inspired has fewer, but richer, diagrams than the UML. While this affects the metric results of individual diagrams it considerably reduces complexity, as fewer diagrams are needed to represent a system.
- The UML has, on average, more definition dependencies per concept.
- IM has a higher average number of successor concepts per concept. This can be attributed to its more flexible diagramming techniques (e.g. Bounding boxes used to group objects for a number of reasons).
- IM has a considerably lower Cyclomatic Complexity Metric (5 compared with 82 for the UML). Therefore, compared to the UML, there is less ambiguity to concept dependencies due to the lower number of dependency paths.

To complement these metrics, the methods are now compared using McLeod's [1997] Method Points analysis.

5.2 Method Points

Method Points measure a modeling language's complexity by comparing models in terms of the generic structure of their deliverables, allowing direct comparison of deliverables under consideration.

Method Points

Diagrams	UML	IM
Use Case diagram	5.5	0
Static Structure Diagram/Business Domain Model	18.5	9
Collaboration Diagram	13	0
Sequence Diagram	6.5	0
Statechart Diagram	9.5	0
Activity Diagram	9	0
Component Diagram	3	0
Deployment Diagram	5	0
Process/Event Diagram	0	17.5
Total	70	26.5

Table 3 - Relative Deliverable Complexity (Method Points)

From the above, it can be seen that, in absolute terms, the UML Class Diagram is twice as complex as its Inspired counterpart, the Business Domain Object Model. Potential explanations for these results include the use of too many links and embellishments in the UML Class Diagram.

The Inspired Business Process Model is slightly more complex than its UML counterpart, the Activity Diagram. The enhanced Inspired Process Model fulfils the functions of several UML counterparts, viz: Use Case, Sequence Diagram, State Chart and Activity Diagram. With the addition of bounding box layers, it also allows modeling Components and Deployment across platforms. While the score for this model is high, the IM introduces various elements of the model progressively in subsequent stages of elaboration as analysis and design proceeds, thus limiting complexity at each iteration to a reasonable level.

The authors would also like to note that the reading and extracting of the various syntax, symbols, links, etc from the UML specification proved most difficult and time consuming. Many points and statements proved to be unclear, ambiguous, inconsistent and confusing. This would lead to difficulty in learning and obtaining the language semantics.

Note too that the IM has made use of the UML notation and diagrams in the creation of the models analysed above. The adaptation deliberately simplified and improved upon these complex models. As can be seen, this aim has been achieved in terms of the Inspired Business Domain Object Model.

6. Conclusions and Further Research

The analysis concluded, with respect to the evaluated methods:

- The addition of RUP to UML substantially assists in providing a more comprehensive method
- The IM provides greater support for strategic alignment, problem determination, contextualizing the problem, business process modeling, and alternative scenario selection than the RUP
- The RUP enables more detailed specification of technical design elements than the IM, such as method specifications and state charts
- The IM is easier to learn and supports higher quality models by constraining the number of concepts it uses, and by reusing these concepts consistently acrosss its models

With respect to the evaluation techniques and process, we concluded:

- Evaluation techniques required considerable effort to apply. In the case of the Krogstie et al framework, there was also a requirement for significant interpretation of terminology.
- The Castellani techniques required some familiarity with directed graph mathematics.
- The way a method is presented can facilitate or hinder its evaluation. The UML specification is difficult to use without significant investment and familiarity. The RUP and IM documents are much more approachable, although more difficult to obtain.
- The NIMSAD framework provides a useful and easily applied technique which provides holistic and thorough examination of methods at a high level

- Complexity metrics are very useful in gauging efficiency of modeling languages and could be more widely used by method designers. The design process should be supplemented by an analysis of the effectiveness of the models, such as that provided by the Krogstie et al analysis

7. References

Bajaj, A. 2000. "SMMM: A Metric Based Framework To Evaluated The Scalability Of Multiple Model Methodologies", *Proceedings of the Fifth* Caise/IFIP8.1 *International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, Stockholm, Sweden.

Brinkkemper, S; Hong,S; Bulthuis, A; Van den Goor, G. 1995. "Object-Orientated Analysis and Design Methods – a comparative Review". (Online) http://wwwis.cs.utwente.nl:8080/dmrg/OODOC/oodoc/oo.html. Accessed 28/04/2001.

Castellani, X. 1998a. "An Overview of the Version 1.1 of the UML Defined with Charts of Concepts", *<<the UML98>> Beyond the Notation – International Workshop,* Mulhouse, France.

Castellani, X., 1998b. "Evaluation of Models Defined with Chart of Concepts: Application to the UML Model", *Proceedings of the Third* Caise/IFIP 8.1 *International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, Pisa, Italy.

De Marco, T. 1979. "Structured Analysis and Systems Specification", *Yourden Press,* New York, USA.

Dori, D. 2000. "Modeling Supply Chain and Electronic Commerce with the Object-Process Methodology", *Proceedings of the Fifth* Caise/IFIP8.1 *International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, Stockholm, Sweden.

Inspired, 2001, Advanced Systems Delivery with Objects, Components, Patterns and Middleware, Course Material, Inspired, Cape Town, South Africa

Jayaratna, N. 1994. *Understanding and Evaluating Methodologies.* McGraw-Hill Book Company, Europe.

Krogstie, J. 2000. "The UML as a basis for the development of models of high quality.", *Proceedings of the Fifth* Caise/IFIP8.1 *International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, Stockholm, Sweden.

Krutchen, P. 2001. *What is the Rational Unified Process?* Rational Software Canada.

Marcos, E., J. Cervera, and L. Fernandez. 1999. "Evaluation of Data Models: A Complexity Metric", *Proceedings of Fourth Caise* /IFIP 8.1 *International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, Heidelberg, Germany.

McLeod, G. 1997. "Method Points: Towards a Metric for Method Complexity", *Proceedings of the Second* Caise/IFIP8.1 *International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, Barcelona, Spain.

McLeod, G. 1998. "Extending the UML for Enterprise and Business Process Modeling", *<<The UML98>> Beyond the Notation – International Workshop,* Mulhouse, France.

McLeod, G. 2001. *Beyond UML: Advanced System Delivery* with Objects, Components, Patterns and Middleware, Inspired Press, South Africa.

Peleg, Dori, 1999. "Experimenting with Real-Time Specification Methods: The Model Multiplicity Problem", *Proceedings of the Fourth* CAiSE/IFIP8.1 *International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, Heidelburg, Germany.

Rational Software Corporation, 1999. Rational Unified Process 5.5, Rational Solutions for Windows, on CD.

Rossi, M. 1996. "Evolution of OO Methods: Unified Case", *Proceedings of First International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, Crete, Greece.

Siau, K. 1997. "Using GOMS for Evaluating Information Modeling Techniques", *Proceedings of Fourth CAiSE/*IFIP 8.1 *International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, Barcelona, Spain.

Siau, K. 1999. "Method Engineering: An Empirical Approach", *Proceedings of Fourth CAiSE/*IFIP 8.1 *International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, Heidelberg, Germany.

Tubio, O. D., Fernandez, M. L., Sanchez-Capuchino, A. M. M. 1999. "On the Capability of Analysis Techniques in Requirements Engineering, *Proceedings of Fourth CAiSE/*IFIP 8.1 *International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*, Heidelberg, Germany.

OMG, 2000. OMG Unified Modeling Language Specification, Version 1.3, First Edition, ww.omg.org accessed April, 2001

Yates, M; Roeleveld, D; Goosen, D, 2001. "A Comparison of the Unified Modelling Language and the Inspired Method", University of Cape Town Empirical Research Study
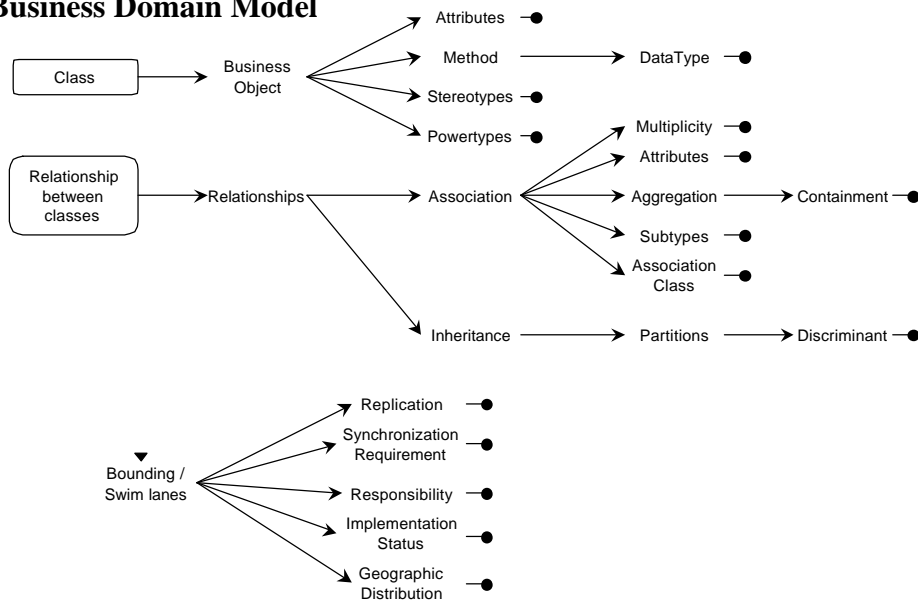
Appendix 1

Method Evaluation Criteria from Literature, Expert Opinion and Informal Survey

Weights: 1 Most Important 5 Inconsequential

Criteria	Importance
The effectiveness in translating business requirements into a system specification	1.2
Ability to handle change in requirements or design during a project	1.5
The ability of each methodology to reach your system modelling goals	1.5
Support alignment of information system view with business strategy/goals	1.7
Communicates ideas and concepts between relevant parties	1.7
Adequacy or completeness in being able to represent a business problem or process	1.7
Representation and integration of business rules into models	2.0
Well defined semantics (meaning and structure of symbols) and notation	2.1
Supports business validation and user feedback (via accessible models, prototypes, etc.)	2.1
Facilitation of easily maintainable evolving systems	2.2
Analyse and design / redesign business processes (in terms of cost, effort, time and risk)	2.2
Suitability to application types (transactional, knowledge based, e-commerce, decision support)	2.2
Ease of progression and development from model to model	2.3
Can be adequately used to model different domains/business problems	2.3
Support standard lifecycles (e.g. iterative, phased or waterfall)	2.3
Is easy to learn	2.4
Adequate representation of people and roles	2.4
Level of industry support	2.4
Level of industry adoption	2.4
Accessible to non-technical business people	2.5
Models can be checked against each other	2.5
Incorporate additional semantics and model expansions	2.5
Ability to support variety of deployment options including desktop, mobile, centralised, distributed, web.	2.6
Availability of supporting tools	2.7
Design user interfaces in terms of look, feel, content and medium	2.7
Caters for modelling and interfacing with existing and legacy systems	2.8
Is concise; no redundancy across models	2.8
Ability to incorporate and support patterns	3.0

Appendix 2 - Inspired Method Charts of Concepts

Business Domain Model

Class → Business Object → Attributes ●
Business Object → Method → DataType ●
Business Object → Stereotypes ●
Business Object → Powertypes ●

Relationship between classes → Relationships → Association
Association → Multiplicity ●
Association → Attributes ●
Association → Aggregation → Containment ●
Association → Subtypes ●
Association → Association Class ●

Relationships → Inheritance → Partitions → Discriminant ●

Bounding / Swim lanes:
- Replication ●
- Synchronization Requirement ●
- Responsibility ●
- Implementation Status ●
- Geographic Distribution ●

Business Process / Event Model

System → Agent → Internal ●
Agent → External ●
Agent → Business Event

Business Event:
- Business Goals → Key Indicators ●
- Application
- Business Object
- Product / Service
- Business Communication

Process → Activity Type
Activity Type:
- Manual
- Computer Supported
- Fully Automated
- Embedded
- Timing

Process → Volume / Frequency ●
Process → Cost ●

Resource Type → Consumption
Resource Type → Constraints
Resource ● ← Resource Type
IO View → Prototype ●
IO Medium ●

Package → Boundary / Swim lane:
- Responsibility ●
- Physical Location ●
- Transaction Process ●
- Deployment Platform ●
- Business Objects Affected ●

Annotations ● ← Rules ← Conditions ← Operators
Synchron. Bar ● ← Probabilities
Synchron. Bar ← Inputs
Events ● ← Triggers ← Outcome
Triggers ← Time