Embedding Requirements within the Model Driven Architecture

Ali Fouad, Keith Phalp, John Mathenge Kanyaru, Sheridan Jeary

Software Systems Research Centre, Bournemouth University, Fern Barrow, Poole, Dorset. BH12 5BB

afouad;kphalp;jkanyaru;sjeary@bournemouth.ac.uk

www.bournemouth.ac.uk/ssrc

***Keywords*:** *Requirements Engineering; Specification; Model Driven Architecture; Business Process Modelling Notation; Use Cases; Computation Independent Model;*

Abstract

The Model Driven Architecture (MDA) brings benefits to software development, among them the potential for connecting software models with the business domain. This paper focuses on the *upstream* or Computation Independent Model (CIM) phase of the MDA. Our contention is that, whilst there are many models and notations available within the CIM Phase, those that are currently popular and supported by the Object Management Group (OMG), may not be the most useful notations for business analysts nor sufficient to fully support software requirements and specification.

Therefore, with specific emphasis on the value of the Business Process Modelling Notation (BPMN) for business analysts, this paper provides an example of a typical CIM approach before describing an approach which incorporates specific requirements techniques. A framework extension to the MDA is then introduced; which embeds requirements and specification within the CIM, thus further enhancing the utility of MDA by providing a more complete method for business analysis.

1.0 Introduction

Business analysts tend to define processes informally, using simple flowcharting notations, whereas software engineers take such informal process notations and add further detail and abstraction to suit the engineering need. The MDA is an approach to software development in which application code can be automatically generated from design models (typically, Unified Modelling Language (UML) class diagrams) (Poernomo et al. 2008). The analysis and design phases are known as the Computation Independent Model (CIM) and the Platform Independent Model (PIM) respectively. A key development activity in MDA is the transformation of a source model to a target model (Phalp et al. 2007). The CIM is the element of the MDA where all requirements and problem domain issues are addressed, however, the CIM is not considered integral to most MDA implementations. For example, the OMG provide a list of "committed companies" (OMG 2007) regarding the architecture, but they themselves neglect the transformation and creation of the CIM (Karow and Gehlert 2006); thus demonstrating the lack of emphasis put on CIM construction.

A notation supported by the OMG for the modelling of business processes is the BPMN (OMG 2005; OMG 2008; White 2004). The BPMN uses software-oriented concepts, similar to those of the UML, which renders the notation inappropriate for the business analyst in defining business process models. Ideally, business users would define workflows in domain specific modelling techniques, which could then be transferable to those modelling techniques used by software engineers in order that code be developed and / or generated directly from models rooted in business logic, rather than software. This has proven to be difficult to achieve due to their differing terminologies, levels of granularity, varied models, approaches, tools and methodologies (Brahe and Bordbar 2006).

Techniques natural to the fields of Requirements Engineering and Business Process Management are not appropriately reflected in the MDA. There is no explicit requirements model, nor is there any connection between Business Process Management and the MDA. The natural evolution of software development requires the convergence of these and other fields to fully connect business and software domains and thus, increase user accessibility to the MDA. However, the current architecture is insufficient for the task. In this paper, one way to bring the MDA closer to the business domain is discussed, that is, including a requirements and specification model within the MDA. Section 2.0 outlines the architecture and discusses the rationale for such an inclusion, followed by an examination of the MDA CIM via a case study in section 3.0. Section 4.0 evaluates the implications for notations and tools by reflecting on the results and section 5.0 outlines a proposed framework that situates requirements theory within the MDA. Section 6.0 summarises the findings and provides conclusions and direction to further work within the area.

2.0 The MDA and Software Requirements

2.1 The MDA Process

The MDA is defined by the MDA Guide Version 1.0.1 (OMG 2003) and is directed at ensuring that what is designed and implemented today, is maintainable and adaptable in the future. The MDA is a conceptual framework and set of standards for a particular software development style (Thangaraj 2004); the focus being to *decouple* the business logic of applications and the underlying technology that provides them (Brown 2004). This separation of interest puts "the business analyst in a unique and potentially powerful position within an organisation" (Slack 2008) because they can affect design and implementation through clear and concise CIM definition. The greatest challenge posed by the MDA appears to be the conflict that exists between what is required by software engineer, that is an abstraction focussed on the definition of software systems, in comparison with that which is required by the business analyst, one rich enough to define business and domain specific processes (Berrisford 2004). It has been seen that "most MDA tools are geared to programmers and software developers rather than non-technical stakeholders" (Kanyaru et al. 2008).

According to the OMG, the MDA has three primary objectives, namely the interoperability, reusability and portability of software artefacts (OMG 2003). The MDA, as opposed to traditional development approaches, "keeps design out of the analysis" (KCL 2002) by specifying different levels of model implementation. Three distinct viewpoints (see Fig. 1) are presented in order to achieve the goals of the MDA by abstracting processes away from computation and technological constraints. They are known as the Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM), and these are the prescribed models to be used within the MDA (OMG 2003).

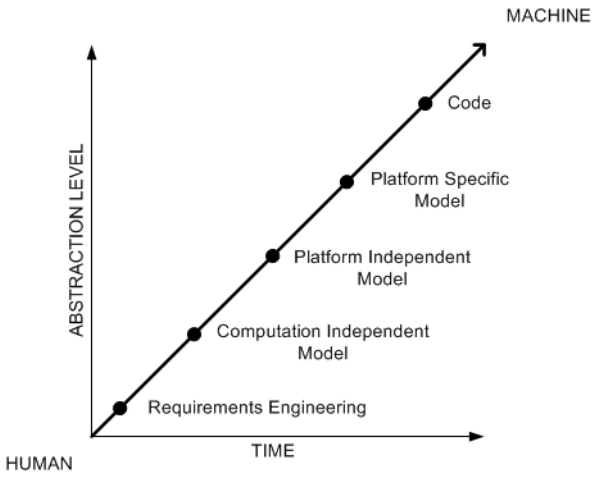

Fig. 1 MDA abstractions on code

The CIM is representative of the “enterprise” and includes defining business rules, facts and terms (Hendryx et al. 2002). However, due to the irregularity of the real world, MDA transformations are focussed on those involving only the PIM and PSM. It is highlighted that not enough emphasis is placed on CIM development under the MDA (Ambler 2007) and the exclusion of the CIM within the MDA transformation process is demonstrated in Fig. 2; no strong connection exists between the CIM and the PIM and there is little research given to transformations involving the CIM, that is CIM-to-CIM and CIM-to-PIM (Kherraf et al. 2008).

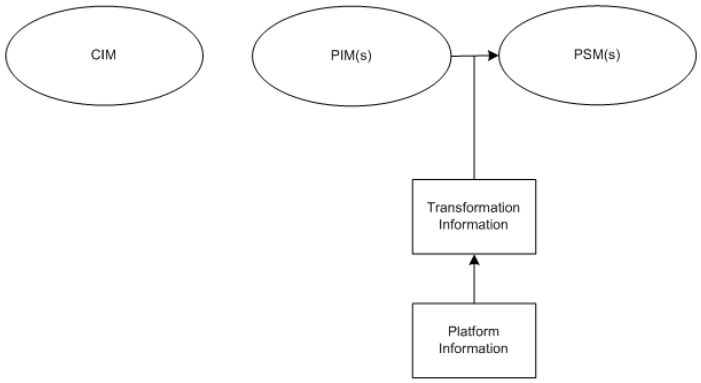

Fig. 2 MDA Viewpoints (Source: developed from (OMG 2003))

There are two opinions when considering upstream transformations. The first is that business process models include enough detail to generate code directly; the second is that business process models should be transformed into computational models, which then facilitate the addition of detail and transformation into code. It is agreed that design cannot be viewed as “a mere transformation of analysis models into software solutions… the input of a transformation has to be complete and all information must be significant for the transformation… by resembling the structure of real world perceptions without regarding software quality requirements, the transformation results are potentially insufficient and unusable in the design process” (Karow and Gehlert 2006).

Requirements Engineering and Business Process Management focus on business processes and associated roles and notations offer freedom to the business user, often allowing an extensibility which simply cannot, or at least is very difficult to, be interpreted by software modelling tools. One way to support the move to design from a process model is to provide a requirements phase where modellers can specify a system that addresses a given business need in a language common to the business user, with resulting systems being akin to customer expectations on quality requirements and the business process. CIM definitions do not currently account for the business users’ unfamiliarity with software development paradigms and mechanisms for transferring requirements knowledge are unavailable.

2.2 Requirements within the MDA

In traditional software development, user needs are defined by requirements, from which a specification of system behaviour is created to address such needs (Jackson and Zave 1995). Once authorised, the specification is transformed into the resulting code for a system to achieve the original user need (Ince et al. 1993). In reality, experience has shown that business processes and requirements are often misunderstood, or even entirely neglected, leaving resulting systems incomplete in meeting stakeholder requirements (Bray 2002; Kappelman et al. 2006; May 1998). This is frequently because developers require business stakeholders to understand requirements and process logic in technical terminology, rather than logic native to business. Insufficient attention has been given to defining requirements in complex projects, those which address unfamiliar problems, and the penalties in terms of cost, quality and the time it takes for projects to be completed have all been well demonstrated; issues pertaining to Requirements Engineering have been found to be instrumental in such failures (Bray 2002; Hansz and Fado 2003). It holds therefore, that time is well spent in defining requirements in the development process. If that time is not invested, then more time and money is spent in fixing problems, rewriting or maintaining erroneous code for missed or incorrectly elicited requirements (Kleppe et al. 2003).

Systems should be delivered that "directly satisfy business requirements" (Koehler et al. 2007). The issue with the MDA is that a requirements model is often neglected. Since, in theory, code is generated as a direct outcome of MDA artefacts, the importance laid upon the CIM is significant. It is here that requirements are first met by computational models and any mistake in the CIM will have ramifications for the implementation unless careful consideration to specification is given.

Traditional models are treated as reference artefacts for subsequent design and coding; MDA models are treated as code. The MDA ensures that modelling *is* software development by taking the level of abstraction a step higher than code. The MDA definition does not prescribe any particular abstraction on the CIM, and therefore neglects this consideration. Stakeholders mistakenly believe that they understand models in the same manner as the software developer, but the truth is, they often have a different understanding (Frankel 2004). To align the paradigms of business and software within the MDA, it is appropriate to use Requirements Engineering and Business Process Management centrally within the framework. UML appears to be the *de facto* standard for the PIM and support is given to the OMG's adoption of the BPMN specification (OMG 2005; OMG 2008; White 2004) for CIM definition. The UML and BPMN are plagued with software engineering concepts and it is difficult to replicate Requirements Engineering concepts using them. The UML in itself may not arguably be the best generic technique for mapping concepts to programming languages, let alone from a business domain into those of the programming world (Génova et al. 2005). Concepts natural to the business domain, such as *role* and *interaction* are not easily represented using UML or BPMN, and that is key.

> "The ability to efficiently design appropriate computer systems and enable them to evolve over their lifetime depends on the extent to which this knowledge can be captured" (Greenspan et al. 1982)

It is the task of the requirements engineer to transform requirements into a richly defined document, suitable for specification, which in turn might be possible to map into the PIM. It is important for the requirements engineer to consider requirements semantics, since much of what is defined as a requirement is in natural English. Despite trying to bridge the gap between the technical developers and the non technical customer, requirements engineers have the added problem in that there are multiple customers, i.e. stakeholders having both a valid, and sometimes invalid, contribution to make. Such contributions need to be managed and refined so that one succinct requirements and specification document might follow. This is why the domain expert is "vitally important" (Jeary et al. 2008). The challenge for the requirements engineer is to provide the business user with sufficient touch-points to communicate in *beginner's* terms and interface to professional software developers.

Since business logic is defined in business models, a research objective is to look at ways of transferring this logic into software models, ensuring that it is represented concisely and consistently in sync with the business model; leading to the eventual generation of specification. It is suggested by the OMG that the primary user of the CIM has inadequate knowledge of software models and concepts (OMG 2003) and therefore, the concentration on the fusion between the business and software engineering worlds is suggested to be the onus of the software developers (Shneiderman 2002). The important thing is for the ordinary business user to be able to understand and apply the MDA, so that requirements are in fact correctly satisfied in the final software product. By challenging the CIM, it is argued that an unambiguous requirements model could be produced at this level, included in the architecture and therefore, heavyweight prototype specifications could be produced, with the added stakeholder benefits of input and understanding. Requirements are effects that are desired in the problem domain. The interface between the problem and application domains forms the system specification (Bray 2002), and it is this that lays foundation to extending the MDA to incorporate Requirements Engineering.

3.0 Case Study: Travel Reservation

In this section, a case study BPMN model is presented as a sample CIM, and requirements are drawn from the notation and presented as a simplistic Use Case diagram for an alternate perspective on requirements and specification. From this, a discussion is provided to review the extent to which a specification model is needed as an intermediate model between Business Process Management and the PIM.

3.1 CIM

The OMG describes the CIM as “the environment of the system, and the requirements for the system” (OMG 2003). The CIM ought to represent the definition of a requirements and specification model that incorporates a true interface by taking elements in the real world, and realising them in the software modelling domain. However, the OMG fails to prescribe a clear definition of the CIM constitution and supports the use of BPMN, and it is this notation that is given focus in this demonstration since it is suggested that the BPMN models fail to match the OMG definition of a CIM.

The objective of the standardised BPMN is two-fold. Firstly, to be understandable to the business community within which it is designed to operate, which is provided for in a simplistic “flow chart” manner to which the business user is already accustomed. Secondly, to be transferable to the software community in a format that is rich enough to be defined and executed. This is challenging and met by the proposal of “mapping” from the BPMN to the Business Process Execution Language or the like, and is how the “BPMN creates a standardised bridge for the gap between the business process design and process implementation” (OMG 2008). It is important to understand that a simplistic business process diagram does not contain sufficient detail for direct mapping for the Business Process Execution Language and therefore, “graphic elements of BPMN will be supported by attributes that will supply the additional information” (OMG 2008). However, the definition of such attributes would in reality likely to have to be completed by the software technical expert, rather than the business analyst, since a complexity is introduced beyond the *Modus Operandi* for business use.

The case study takes the form of a simplistic travel reservation system; the BPMN model for this system is shown in Fig. 2.

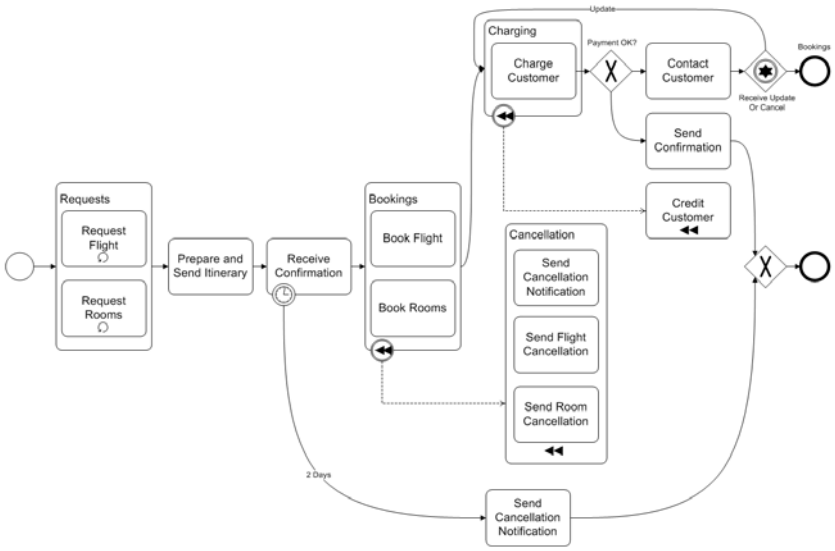

Fig. 2 Travel reservation system in BPMN (source: adapted from (Silver 2008c))

In summery, the demonstration CIM involves a fictional scenario (which is based on Silver's work (Silver 2008c)) and represents a travel reservation system whereby flight and hotel details are input, itineraries created and verified, payments accounted for, and holidays are booked with the travel agent. This is a simplistic process (outlined in a single paragraph of natural English) yet involves a somewhat convoluted and confusing diagrammatic definition via the BPMN. Moreover, because the components of the BPMN are derived from software engineering semantics, the accuracy and correct understanding of the notation is imperative in the definition. In the following section, the same demonstration is accounted for within a simplistic technique for specification, with further discussion and analysis beyond this given in section 3.3.

3.2 Requirements and Specification Models

Business analysts "do not start with textual requirements: it is too complex to come out right away with them. Instead, they usually start with simple graphic diagrams" (Rivkin 2008). In this section, the BPMN outlined in the previous section is used to produce a set of Use Case specifications. Components are broken down into the relevant *users* of the system and the *tasks* they complete during their interactions with the system (Stevens and Pooley 2000). "Use Cases are part of UML and offer a foundation or starting point for using models" (Hansz and Fado 2003). Use Case diagrams are useful in Requirements Engineering, mainly to capture which actors are required to interact with the system, and for each, the specific tasks with which they interact (Stevens and Pooley 2000). However, Use Cases are not mainstream artefacts of the MDA, nor are they "suitable for code generation or model execution" (McNeile 2003). A Use Case diagram is presented in Fig. 3, along with a matching Use Case description in Table 1.

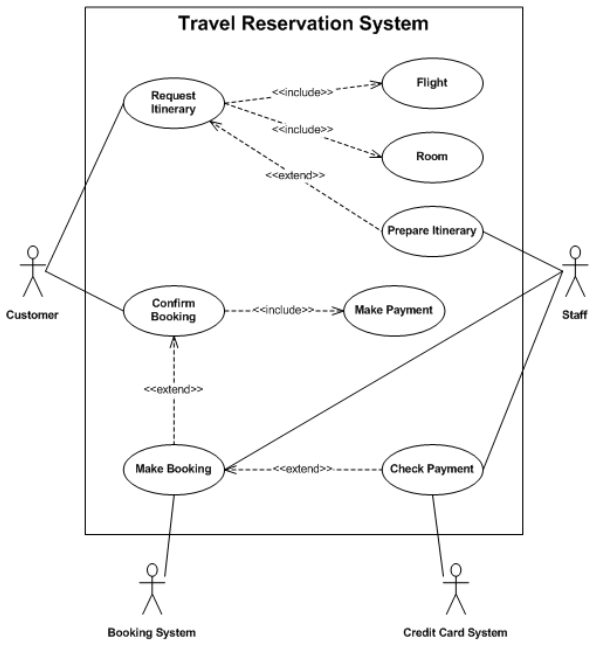

Fig. 3 Travel reservation system represented by a Use Case diagram

Use Case Title: Travel Reservation System

Actors: Customer, Staff, Credit Card and Booking System

Context / Purpose: To enable the booking of a holiday

Pre-Condition: Staff is available to accept itinerary request

Event Flow:

1. Customer requests Itinerary from Staff
2. <<includes>> Customer selects flight details
3. <<includes>> Customer selects room details
4. Staff prepares Itinerary
5. Staff sends Itinerary to Customer
6. Customer sends Confirmation to Staff
7. <<includes>> Customer makes Payment
8. Staff makes Booking
9. Staff checks Payment with Credit Card System (Constraint: Credit Card System is available)
10. Staff sends Confirmation

Alternatives:

Alternatives to this Use Case could be that the itinerary fails to reach the customer or the customer fails to update staff regarding payment problem. However, such an alternative has been ruled out due to insufficient elicitation and will be discussed in the subsequent section.

11. If Confirmation fails, Staff sends Cancellation Notice to Customer (Constraint: Wait 2 days)
12. If Payment fails, Staff notifies Customer
13. Customer updates Staff
14. Customer cancels Booking

Exceptions:

Exceptions to this Use Case could be that there is a system-wide failure; the credit card or booking system is unavailable. However, such exceptions have been ruled out due to insufficient elicitation and will be discussed in the subsequent section.

15. If Booking fails, Staff sends Cancellation Notice to Customer
16. Staff cancels Flight and Room with Booking System (Constraint: Booking System is available)
17. Staff sends Cancellation Notice to Customer
18. If Charging fails, Staff credits Customer

Post-Condition: Booking confirmation or cancellation notice is sent to the customer

Table 1 Travel reservation system represented by a Use Case Description

3.3 Discussion

The BPMN is primarily a software notation. The use of a software-oriented notation for the purpose of process modelling does not provide models which are accessible to domain experts. It is illogical for business to utilise “diagramming techniques that are used by software developers since they are designed to notate and model all manner of things that business managers don’t need to be concerned with in order to manage the business” (Harmon 2005).

Initially, it became evident that there would be some difficulty in distinguishing elements that were important and required by the system, from those that were not. The BPMN model has no such system boundary view, and it is unclear as to whether the customer or staff member would enter the initial booking request, which is clarified with examination of the Use Case diagram.

In the BPMN demonstration, notions of *role* or *actor*, and *interaction* do not exist at all, and thus do not transfer into the Use Case diagram and description. This highlights the insignificance that BPMN places on such notions, which are of course natural to the business world and requirements methods. The Use Case definition provides a base which is altogether different in comparison to the BPMN model. Although many identified cases remain the same or similar to the BPMN tasks, the Use Case was found to be better aligned with the reality of the situation in hand. Roles, interactions and collaborations between parties are easily identifiable, for example, the “check payment” case involving the staff member and the credit card system.

The important thing to an ordinary business user is to be able to understand, and apply the CIM, so that requirements are in fact met in the final software product. The Use Case is viewed as a naturally simplistic tool, and therein lies the beauty; being useful to communicate to the heart of the business user, rather than introducing the complexities that the BPMN has to offer and, although simplistic in nature, the Use Case specification is becoming “increasingly integrated with model elements” (Hansz and Fado 2003) of the MDA.

From the Use Case specification, it is clear to see that the travel reservation system employs a deal of human interactivity. The BPMN solution however presents a wholly computerised, sequential view of that interactivity at a particular level of abstraction. Because of its nature, “BPMN does not match the reality of human behaviour” (Harrison-Broninski 2006a) and therefore is unable to model human-driven processes efficiently.

A mechanism was not available to incorporate non-functional requirements within the context of the BPMN diagram. For example, a specification might request that the designed system be compatible with the existing system (such as the credit card and booking systems). This is a platform specific consideration that is beyond the current scope of the CIM and PIM. However, this can easily be accommodated within the supporting documentation of constraints in Use Case descriptions to ensure such non-functional requirements are carried through beyond the CIM.

The eventual system is likely to issue an email address, or perhaps SMS functionality, to cater for the *Send* and *Receive* requests between the staff and customer. This requirement is not specified in BPMN; it is only from the analysis of the Use Case description that this issue is raised.

Alternative and exceptional Use Case discovery is part of Use Case exploration. On review, it is clear that the BPMN contains no indication to what might happen should the itinerary fail to reach the customer, the customer fail to update the staff regarding a payment problem, a system-wide failure be experienced or the credit card and booking systems be unavailable. Exceptions are delivered in the BPMN by relating to an event that might require the cancellation of a booking or an error requiring the customer be credited for amounts charged. Use Cases naturally raise such questions as cause for concern to be verified by the stakeholder involved and provide sufficient elicitation from a Requirements Engineering perspective in the construction of the specification of a software system.

Techniques such as the BPMN are unsuitable for defining human behaviours and biased to supporting technological implementations (Harrison-Broninski 2005c). Put simply, business and software analysts think they understand the BPMN in the same manner (Silver 2008a)), but the truth is, the specification can be understood quite differently by those involved. From reviewing samples of BPMN distributed in training materials, significant errors were found and highlighted in (Silver 2008d; Silver 2008e). Semantics relating to sequence flow, gateway attributes, intermediate events, sub-process boundary protocols, compensation events, transaction sub-processes, cancel events, link events, state based synchronisation etc are all illustrative of the complexities involved with the BPMN. If the tools and teaching materials cannot get the notation right, it is difficult to see how business users may make anything more of the BPMN than simple flowcharting. It is agreeable that “there isn’t much educational material out there that shows people how to use BPMN correctly” (Silver 2008d), however, once plugged into the MDA, CIM level errors could be a critical project management concern.

4.0 Implications for the MDA

4.1 Notations

The BPMN was designed to make the implementation of executable models easier, driven by one notation, rather than be flexible to the multitude of notations. It is said that the BPMN provides “the power to depict complex business processes and map to Business Process Management execution languages” (OMG 2008), however, limiting the modelling of the business domain to a single notation provides a weakness in that concepts which could have been adopted in other notations may no longer be used. Moreover, by definition, if BPMN is based on software concepts, the technique is no longer as meaningful to the business analyst as it is to the software producer, leading the business analyst to learn to think, for example, in terms of object rather than process.

The definition is semantically rich, whereby each process is defined using nodes and connections that have specific meaning, and it is those semantics that have the interest of the IT world. The notation can indeed be used to convey ideas of surface flowcharting, but to the orthodox user, semantics can be drawn for such sketching which may result in misunderstandings, especially in terms of the MDA and the CIM. The BPMN is a notation that can only "vaguely" represent the business process (Harrison-Broninski 2006a) due to the this software-orientation.

One solution might be to open out the MDA and allow the accessibility of a greater range of modelling notations and standards support, rather than giving focus to just the chosen few. The BPMN and the UML are not intended to be supportive or facilitate domain specific modelling definitions in terms of the business analyst (Brahe and Bordbar 2006). A greater acceptance of other notations and standards support might enable the MDA to fulfil the ideology of facilitating truly interoperable, portable and reusable models. For example, the BPMN only allows messages to interact with two single entities, whereas with human-driven processes, interaction is usually between many (a conference call, for instance), which is allowed using other diagrammatic notations such as the Role Activity Diagram. As previously mentioned, the BPMN is not capable of producing a *Role*, and therefore, requirements pertaining to such a notational element cannot be modelled correctly for that reason and systems are delivered incorrectly. In the BPMN, roles have been aligned to a grouping of activities or process chains via *swim lanes* and *pools*. Roles are essentially private information spaces, accounting for the responsibility and purpose that characterise the people and systems that they represent, and not just a collection of mechanised actions and functions; Role Activity Diagrams have a simple foundation, with a "natural" graphical notation, which allows them to be introduced to the business user with minimal training (Harrison-Broninski 2005a).

An alternative is to examine transformation techniques in the MDA, specifically those relating to CIM-to-CIM and CIM-to-PIM. One technique could be to transform Role Activity Diagrams to the UML for the use in software systems and the MDA. Harrison-Broninski outlines formally how each component in a Role Activity Diagram can be represented in the UML, although with less "specialised usability", and concepts pertaining to Role Activity Diagrams and Human Interaction Management tools (Harrison-Broninski 2005b). Other researchers also focus on such transformations. A proposed methodology for transforming models of the BPMN into Use Case notation goes some way to looking beyond the OMG's CIM and towards the PIM (Rodriguez et al. 2007). Secure Business Processes are defined by using an extension of the BPMN Business Process Diagram and the Business Process Security Profile. The OMG's Query / View / Transformation rules are then used to capture the BPMN and transform it into elements within UML Use Cases. For example, *Pool* to *Actor*, *Activity* to *Use Case* and *Security Requirement* to *Use Case* (Rodriguez et al. 2007).

Ultimately, the task of retaining the richness contained within requirements models that may be lost in a transformation process remains a difficult one unless the requirements model is held

integral to the MDA. The BPMN does not include the technology suitable to retain the richness of information that is provided for by requirements models. An alternate suggestion for investigation could be to include, or extend the notations of the BPMN to account for notions that carry real meaning in requirements models. For example, as previously discussed, the “Swim lane” notation could be adapted to reflect that of a “Role”. However, this notion is dismissed in other research, suggesting that such richness simply cannot be captured by the BPMN via extension (Harrison-Broninski 2006b).

4.2 Tools

MDA tools and processes are required to be agile enough to adapt and facilitate changes. Models allow ideas to be “shared in abstractions” (Soley 2006). In order for the MDA to be successful within industry, tools must be accessible to both the software developers that create implementations and the business users whose requirements necessitate them. Many tools are available in the market and purport to pursue MDA ideals. However, it has been seen that “most MDA tools are geared to programmers and software developers rather than non-technical stakeholders” (Kanyaru et al. 2008) and that such tools are still in an evolutionary state (Leonardi and Mauco 2004).

The MDA is conceptually very simple to understand, yet complex to implement. To make the MDA work there is a requirement to go beyond MDA modelling tools, such as Bridgepoint, TogetherCC and OptimalJ (Ambler 2007). It is written that the MDA does not include “precise rules or guidelines explaining how software engineers can use” the CIM, PIM and PSM (Garrido et al. 2007). The only example of MDA in action is from the J2EE platform for PIM to PSM mappings. This leads vendors (such as Microsoft) to bend the rules on the usage of strict guidelines (such as the UML) to support .NET mappings (Frankel 2004). An issue that has been discussed in academia is that the central notion that the PIM is flawed since tools focus on particular methods and do not allow for model and tool integration. Some modelling languages are simply better at managing specific concepts then others and therefore, to truly realise the MDA, some interoperable solution between model transformation languages is required since “no single language can be adapted to all application domains” (Jouault and Kurtev 2006).

Eclipse is an open source Integrated Development Environment that supports the MDA. Central to Eclipse is the Eclipse Modelling Framework, which is a “model-driven metadata management framework” (Frankel 2005). In marketing Eclipse to industry, the unique selling point is highlighted to be the modelling and code generation capabilities, little focus is given to the consistency and automation of the Integrated Development Environment; which is the real advantage (Frankel 2005). Perhaps this is because the Integrated Development Environment capabilities are too complex and difficult to market to business users. Indeed, by giving attention to modelling and code generation, interest can be gained by organisational managers and investments in technologies can be made. However, if the market is purely represented by

technologists, then this should not be the case. In reality, Eclipse does not have the necessary tools to generate applications such as *Purchase Order Processing*, "it is better suited… for modelling a tool's metadata and generating code that manages the metadata" (Frankel 2005). The Eclipse marketing strategy therefore, could be considered to be aimed at increasing the business interest, without any real promise to those users, delivering directly to the technologists working behind the scenery.

As previously highlighted, the BPMN has associated problems. Difficulties are also apparent in industry in that the Business Process Management field has not yet matured. Tools are available, COTS and custom created, to accomplish all manner of process analysis, design and execution tasks. It is noted that business users are not embracing the BPMN to the degree that the OMG and MDA proponents would hope for. One reason suggested for this is that "the tool vendors themselves don't follow the spec" (Silver 2008b). It has been highlighted that well known distributors of BPMN software tools and training facilities (analysis derived from a review that includes Savvion, Tibco, Appian and Intalio's Open Source Modeller) neglect the BPMN specification (Silver 2008b; Silver 2008d). Supporters of BPMN therefore must work tirelessly to ensure that tools and training integration incorporate the correct definition of the BPMN standard as prescribed by the OMG (OMG 2008). It is also noted that "modelling tools obviously don't include validation routines that weed out illegal diagrams" (Silver 2008b), which may also benefit consideration. Formal semantics could be defined and, via automation, rules could be generated and model semantics verified which would "ensure precise specification and… assist developers in moving towards correct implementation of business processes" (Wong and Gibbons 2008). However, the technique, supported by Microsoft research, draws the definition of semantics in CSP via a mathematical state base in the notation Z. The mathematical base and process algebra syntax is very complex and unhelpful for the business user, in definition and verification of system requirements (Wong and Gibbons 2008). Requirements and testing experts "often stay outside the modelling tools. They need to be brought in" (Hansz and Fado 2003). From the perspective of the MDA, they are currently outside of the architecture as a whole, beyond the tooling task.

Inclusive agile methods, which are akin to those of Requirements Engineering, are proposed to be used to provide non technical stakeholders with sufficient tools that are simple to understand and can adequately transfer into a technical PIM (Ambler 2007), but are not widely applied. The Business Process Developing Life Cycle is defined as an approach to eliminate the business / system analyst knowledge gap, by formalising business requirements in terms of a Basic Business Process Flow (Rivkin 2008). Another solution is to adapt an interactive environment (perhaps using wizards and transformations between business and MDA artefacts) that eases the business user into specification within the MDA process, without getting too involved in the technicalities related to the architecture. One such tool developed to facilitate the accessibility of the MDA to the business user is presented in the Visualise All Model Driven Programming initiative (VIDE 2007). The tool "provides a development environment" (VIDE 2007) consisting of several palettes, guiding the user from pre-CIM to PIM. The tool represents a useful step in the right direction of

making the MDA accessible to the business user. Another drive to CIM specification is the Topological Functioning Modelling for Model Driven Architecture, via UML Use Cases and Conceptual Class Diagrams (Osis et al. 2007). The underlying architecture is that "functionality determines the structure of the planned system" (Osis et al. 2007), however, to limit the definition in such a way perhaps neglects some important pre-CIM and design considerations that should be made in the derivation of a suitable concept tool.

The focus for MDA vendors is to "provide integration with requirements and testing artefacts within the tool… if models can be tested against accurate specifications, then problems are caught upstream where they can be handled" (Hansz and Fado 2003).

5.0 Solution Framework

So far, the importance of including the Requirements Engineering paradigm within the MDA has been established. Cost savings and an early release date can be achieved by optimising the time spent in software development. The MDA is broad enough to accommodate many methodologies from the common waterfall approaches to more recent methods such as Agile Software Development and Extreme Programming (Kleppe et al. 2003; OMG 2003). With an increasing demand for systems to match the requirements prescribed by business managers at the start of a project, there is an even greater need for projects to reflect multiple changes throughout the development process. Perhaps the most defining characteristic of the MDA is conceptual simplicity. The MDA fosters an agile environment which goes some way to addressing the concern that development projects are not reactive enough to the changing environments. Developers have been using models for years, business users even longer. The UML is now the operating standard to which MDA is associated, facilitating transformations between models with the ability to be flexible to changing requirements.

As noted, upstream transformations remain relatively unsuccessful and there is no extensive study of this. Those that attempt the task report a loss of richness in process models. Other researchers look to incorporate requirements within the MDA. In (Poernomo et al. 2008), a methodology is suggested that can react to requirements changes, "accurately reflect them to code and ensure that the successful completion of the IT system will add value to the business" (Poernomo et al. 2008), however no verifiable data is provided and the solution seems conceptually too complex for business to adopt. The CIM-to-PIM transformation process is inadequately described and the resultant diagram is not necessarily a PIM, since no account for design has been made. In (Martin and Loos 2008), an integration language based upon the BPMN is discussed to provide both the simplicity to the business user and enough complexity to the software engineer in order that programs might be created "automatically" (Martin and Loos 2008). This is difficult because by introducing automatic transformations from the CIM to code constrains the CIM and renders the PIM redundant. Design features are shifted into CIM construction, leaving a large part of the development process down to non-technical business users and the BPMN is complex; it may not

be the best notation for business users to demonstrate business processes, let alone design and implement software solutions. In (Kherraf et al. 2008) a process is identified and demonstrated via a case study. Although significant in furthering the understanding of the value and feasibility of such transformations, no consideration is given to the fact that analysis and design models are from two opposing environments. As previously noted, directly transferring detail from an analysis model does little more than create an object oriented view of the problem domain (Génova et al. 2005), by defining systems in such a way that might involve automatic transformations, design decisions are imposed on *upstream* models, of which ought to be created by business, not system analysts, which is not addressed.

A solution is therefore suggested in extending the MDA framework with the Requirements Engineering paradigm and is concerned with solving two overarching problems. Firstly, the need to provide a software process structure into which different uses of requirements and specification can be accommodated, and secondly, the need to provide a standard set of techniques for expressing requirements and specification within the MDA. Fig. 4 illustrates how the MDA can be extended via a CIM that appropriately situates Requirements Engineering within the MDA and connects the CIM to the PIM via a technique that ensures the distinction between the problem and solution domains.

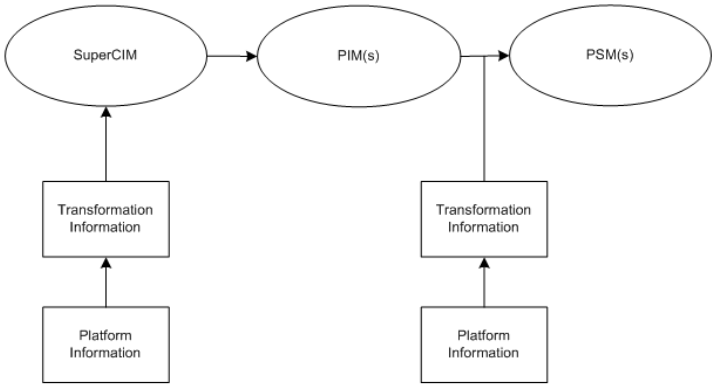

Fig. 4 Extended MDA (Source: developed from (OMG 2003)

In software development, “attention is focussed at the application level, where the emphasis is on achieving desired functionality” (Beeson et al. 2002). The MDA prescribes an approach that should be simple for non-technical stakeholders to provide and produce artefacts that are understandable in nature to that stakeholder (Soley 2006). However, PIMs and PSMs both have a complex nature and must contain enough detail to generate the associated code. Moreover, the inclusion of a CIM that focuses only on software languages such as the UML or BPMN is unhelpful, and leads to the neglect of requirements and specification, giving focus only to such functionality. It is argued here that, for the transformation of the CIM to be useful post-CIM and to adequately make the connection to the software domain, it must be representative of a traditional specification, and not an abstraction on the problem domain. The system boundary distinguishes those functions required by the system from those that are not; and is important in the

determination of requirements because it defines where the software element is situated in comparison to the overall system (Nuseibeh and Easterbrook 2000). The abstraction is a fine line between interactive human and software elements and, when working with a PIM level language such as the UML, the imposition of object orientation compounds this elemental distinction. The system boundary is currently not an explicit consideration of the CIM and this is accounted for in the SuperCIM by manually transferring only the system elements into the software domain for realisation in design. This is achieved by applying information about the transformation (i.e. mapping rules) that is informed by the target platform meta-model (i.e. the UML) and extracting elements in the process. This represents a true interface by taking elements in the real world that are to be realised in the software domain and transferring them into it, allowing for traceability to be accounted for in the documentation trail.

Ultimately, the goal is to facilitate MDA accessibility by allowing business users to define their requirements in *any* format to which they can understand and apply; which retains the versatility of the MDA in being interoperable, portable and reusable. This then must be matched to a SuperCIM representation that is sufficient for the derivation of software systems requirements in a manner to which the software engineer is akin to and also retains the richness of definition provided by the business user. The CIM must embrace ideals from both the world of Business Process Management and Software Systems Engineering, and these ideals are somewhat conflicting. In software Systems Engineering ‘Dijkstra structures’ are used where everything is neatly structured; the real world is much less coherent and applying such modelling techniques to business processes is inadequate (Ould 2004). Furthermore, there is no sense in modelling human behaviours, collaborations and interactions that occur in the business process, but have no significance within software systems. So, not only should the CIM be computationally independent (a business process model alone in fact will not do), it must also be independent of human processes that are not to be realised in the final system, since inclusion of such items will lead to ambiguities and unnecessarily complex models. The CIM must facilitate modelling of real world disorder whilst not neglecting notations to which software engineers are accustomed to; the CIM is the perfect candidate for a gateway between the ‘real’ and ‘software’ worlds via specification, and ultimately delivering software systems in satisfaction of such real life requirements.

6.0 Summary

A case study scenario has been presented as a sample artefact of the MDA, with further analysis representing the case study as specification. The study demonstrates how complexities relating to the BPMN may not be beneficial to the business analyst in expressing the requirements of software systems, especially in application of the MDA CIM. The BPMN provides an inadequate representation of human behaviour (Harrison-Broninski 2005c) and the UML and MDA code generators are “not the panaceas that some would have us believe” (Thomas 2004). The majority of developers have not yet begun sketching with the UML, let alone developed the art of creating sophisticated models using such tools that are required by the MDA (Uhl and Ambler 2003). The

lack of definition from the OMG as to what precisely makes a CIM leaves open the door for the application of any number of techniques, the challenge for MDA begins with bringing together very different worlds in the abstraction of an appropriate CIM.

The fusion of business and software processes is an important step for the MDA and one that will need to be made before infrastructural consolidation is made. There is a need for the CIM definition to adequately account for requirements, without any constraint on modelling method or tool, in order that such fusion between business and software be realistic (Hansz and Fado 2003). The suggestion for extending the MDA is identified; supporting specification without any loss in the versatility of the MDA, providing a conduit to interoperable, portable and reusable software models. This is suggested to be achievable by extending the MDA to account for requirements and specification via precise CIM construction using transformation and platform information to deliver an appropriate CIM from defined requirements into the software domain by a systems analyst knowledgeable in both the business and software processes.

6.1 Further Work

The presented ideas could be supported by further trials, perhaps using alternate modelling techniques, such as Role Activity Diagrams. Research into how different a PIM might be defined, using such notations, may hold some value in the future of the CIM. It is suspected that mechanistic and humanistic processes might both be a requirement for Business Process Management, but only those that can be mechanised for the MDA. Either way, an understanding of the nature of business processes is central, since humanistic processes presented incorrectly at the CIM level may result in a deformed architecture which does not facilitate support of the business process it was defined to augment. The notion could also be applied to large scale systems, with the development of a concept tool to facilitate requirements and specification within the MDA. In a large portfolio with many requirements, dependency might become an issue where such requirements are all aligned with their respective constraints and conditions and a situation may occur whereby any alteration will force a change or infringe on such a constraint or condition. The suitability therefore of the extending the MDA in enterprise systems requires further investigation.

The real benefit to implementing a requirements and specification model within the CIM is that it ensures the interactive collaboration between non-technical stakeholders and technical developers in generating requirements for the system to be delivered in the PIM, rather than transferring elements from the problem domain into a software systems design model. It is likely to remain difficult to make a pronounced connection with business with the current deficiency of real requirements and specification within the MDA.

7.0 References

AMBLER, S. W. (2007). A roadmap for agile MDA. Resource Document. http://www.agilemodeling.com/essays/agileMDA.htm. Accessed 26th October 2007.

BEESON, I., GREEN, S., SA, J., & SULLY, A. (2002). Linking Business Processes and Information Systems Provision in a Dynamic Environment. *Information Systems Frontiers*, 4 (3), 317-329.

BERRISFORD, G. (2004). Why IT Veterans Are Sceptical About MDA. *Second European Workshop On Model Driven Architecture*. Canterbury, UK.

BRAHE, S., & BORDBAR, B. (2006). A Pattern-Based Approach to Business Process Modeling and Implementation in Web Services. *Service-Oriented Computing, ICSOC 2006, 4th International Conference Workshop Proceedings.* Chicago, IL, USA.

BRAY, I. (2002). *An introduction to requirements engineering.* England: Addison Wesley.

BROWN, A. (2004). An Introduction To Model Driven Architecture. Resource Document. http://www-128.ibm.com/developerworks/rational/library/3100.html. Accessed 20th November 2007.

FRANKEL, D. S. (2004). MDA Journal: Domain Specific Modeling and Model Driven Architecture. Resource Document. *A BPT Column.* http://www.bptrends.com/deliver_file.cfm?fileType=publication&fileName=01%2D04%20COL%20Dom%20Spec%20Modeling%20Frankel%2DCook%2Epdf. Accessed 6th January 2004.

FRANKEL, D. S. (2005). MDA Journal: Eclipse and the MDA. Resource Document. *A BPT Column*. http://www.bptrends.com/publicationfiles/03%2D05%20COL%20Eclipse%20and%20MDA%20%20%2D%20Frankel1%2Epdf. Accessed 24th September 2008.

GARRIDO, J. L., NOGUERA, M., GONZLEZ, M., HURTADO, M. V., & RODRÍGUEZ, M. L. (2007). Definition and use of Computation Independent Models in an MDA-based groupware development process. *Science of Computer Programming*, 66 (1), 25 - 43.

GÉNOVA, G., VALIENTE, M. C., & NUBIOLA, J. (2005). A Semiotic Approach to UML Models. *First International Workshop on Philosopical Foundations of Information Systems Engineering (PHISE 2005) - In the proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE 2005) workshops.* Porto, Portugal.

GREENSPAN, S. J., MYLOPOULOS, J., & BORGIDA, A. (1982). Capturing more world knowledge in the requirements specification. *Proceedings of the 6th international conference on Software engineering.* Tokyo, Japan: IEEE Computer Society Press.

HANSZ, D., & FADO, D. (2003). Unambiguous, Non- Binding Requirements for MDA. *MDA™ Implementers' Workshop Succeeding with Model Driven Systems.* Burlingame, CA, USA

HARMON, P. (2005). Email Advisor: Standardising Business Process Notation. *Business Process Trends*, 3 (19).

HARRISON-BRONINSKI, K. (2005a). Modeling Human Interactions: Part 2. Resource Document. *BPTrends.* http://www.businessprocesstrends.com/publicationfiles/07%2D05%20WP%20Modeling%20Human%20Interactions%20%2D%20Pt%202%20%2D%20Harrison%2DBro%E2%80%A6%2Epdf. Accessed 29th August 2008.

HARRISON-BRONINSKI, K. (2005b). RADs And The UML. Resource Document. 1.0 http://human-interaction-management.info/RADs_and_the_UML_1_0.pdf. Accessed 2nd September 2008.

HARRISON-BRONINSKI, K. (2005c). The Technology Of Human Interaction Management. Resource Document. *BPM.COM*. http://www.bpm.com/FeatureRO.asp?FeatureId=168. Accessed 26th March 2008.

HARRISON-BRONINSKI, K. (2006a). BPM, Anyone? Resource Document. *BPTrends.* http://www.businessprocesstrends.com/deliver_file.cfm?fileType=publication&fileName=02-06-DIS-Re-ReStandardizingBPNotation-Harrison-Broninski.pdf. Accessed 6th February 2008.

HARRISON-BRONINSKI, K. (2006b). The Future Of BPM (Parts 1 to 6). Resource Document. *BPTrends.* http://www.businessprocesstrends.com/resources_publications.cfm?publicationtypeID=DFFB9D1C-1031-D522-3AAF1211DDD4AD95. Accessed 18th November 2008.

HENDRYX, S., VINCENT, P., & CRIBBS, J. (2002). Business Rules with MDA. *Third Workshop on UML® for Enterprise Applications: Model Driven Solutions for the Enterprise.* San Francisco, CA, USA: OMG.

INCE, D. C., SHARP, H., & WOODMAN, M. (1993). *Introduction to Software Project Management and Quality Assurance.* Maidenhead, Berkshire, England: McGraw-Hill, Inc.

JACKSON, M., & ZAVE, P. (1995). Deriving specifications from requirements: an example. *Proceedings of the 17th international conference on Software engineering.* Seattle, Washington, United States: ACM.

JEARY, S., FOUAD, A., & PHALP, K. (2008). Extending The Model Driven Architecture With A Pre-CIM Level. *the 1st International Workshop on Business Support for MDA, co-located with TOOLS EUROPE 2008.* Zurich, Switzerland.

JOUAULT, F., & KURTEV, I. (2006). On the architectural alignment of ATL and QVT. *Proceedings of the 2006 ACM symposium on Applied computing.* Dijon, France: ACM.

KANYARU, J. M., JEARY, S., COLES, M., PHALP, K., & VINCENT, J. (2008). Assessing Graphical User Interfaces In Modelling Tools For MDA Using Cognitive Dimensions. *Software Quality Management 2008.* University of Ulster, Newtownabbey, Northern Ireland: The British Computer Society.

KAPPELMAN, L. A., MCKEEMAN, R., & ZHANG, L. (2006). Early Warning Signs of it Project Failure: The Dominant Dozen. *Information Systems Management: IT Project Management*, 23 (4), 31 - 36.

KAROW, M., & GEHLERT, A. (2006). On the Transition from Computation Independent to Platform Independent Models. *Proceedings of the 12th Americas Conference on Information Systems (AMCIS 2006).* Acapulco, Mexico.

KCL. (2002). Supporting Model Driven Architecture With eXecutable UML. Resource Document. http://www.kc.com/download/index.php. Accessed 26th June 2008.

KHERRAF, S., LEFEBVRE, E., & SURYN, W. (2008). Transformation from CIM to PIM Using Patterns and Archetypes. *Proceedings of the 19th Australian Conference on Software Engineering (ASWEC 2008). 25 - 28 March 2008*, Perth, Western Australia: IEEE Computer Society.

KLEPPE, A. G., WARMER, J., & BAST, W. (2003). *MDA Explained: The Model Driven Architecture: Practice and Promise.* Addison-Wesley Longman Publishing Co., Inc.

KOEHLER, J., GSCHWIND, T., KÜSTER, J., PAUTASSO, C., RYNDINA, K., VANHATALO, J., et al. (2007). Combining Quality Assurance and Model Transformations in Business-Driven Development. *Third International Workshop and Symposium on Applications of Graph Transformation with Industrial Relevance (AGTIVE 2007).* Kassel, Germany.

LEONARDI, M. C., & MAUCO, M. V. (2004). Integrating natural language oriented requirements models into MDA. *Anais do WER04 - Workshop em Engenharia de Requisitos.* Tandil, Argentina.

MARTIN, A., & LOOS, P. (2008). Software support for the Computation Independent Modelling in the MDA context. *the 1st International Workshop on Business Support for MDA, co-located with TOOLS EUROPE 2008.* Zurich, Switzerland.

MAY, L. J. (1998). Major Causes of Software Project Failures. *Crosstalk - The Journal Of Defense Software Engineering*, 9 - 12.

MCNEILE, A. (2003). MDA: The vision with the hole? Resource Document. http://www.metamaxim.com/download/documents/MDAv1.pdf. Accessed 25th October 2007.

NUSEIBEH, B., & EASTERBROOK, S. (2000). Requirements engineering: a roadmap. *Proceedings of the Conference on The Future of Software Engineering.* Limerick, Ireland: ACM.

OMG. (2003). MDA Guide Version 1.0.1. Object Management Group.

OMG. (2005). BPMI.org and OMG Announce Strategic Merger of Business Process Management Activities. Resource Document. http://www.omg.org/news/releases/pr2005/06-29-05.htm. Accessed 21st January 2009.

OMG. (2007). Committed companies and their products. Resource Document. http://www.omg.org/mda/committed-products.htm. Accessed 25th October 2007.

OMG. (2008). Business Process Modeling Notation, V1.1. Object Management Group.

OSIS, J., ASNINA, E., & GRAVE, A. (2007). Formal Computation Independent Model Of The Problem Domain Within The MDA. *10th International Conference on Information System Implementation and Modeling. ISIM, 07.* Hradec nad Moravicí, Czech Republic: IEEE.

OULD, M. A. (2004). *Business Process Management: A Rigorous Approach.* British Computer Society.

PHALP, K. T., JEARY, S., VINCENT, J., KANYARU, J. M., & CROWLE, S. (2007). Supporting Stakeholders in the MDA Process. *Software Quality Management / INSPIRE 2007*. University of Tampere, Finland.

POERNOMO, I., TSARAMIRSIS, G., & ZUNA, V. (2008). A methodology for requirements analysis at CIM level. *the 1st International Workshop on Business Support for MDA, co-located with TOOLS EUROPE 2008.* Zurich, Switzerland.

RIVKIN, W. (2008). Closing the Business-IT Gap Once And For All. Resource Document. *BPMInstitute.ORG*. http://www.bpminstitute.org/articles/article/article/closing-the-business-it-gap-once-and-for-all.html. Accessed 28th January 2009.

RODRIGUEZ, A., FERNANDEZ-MEDINA, E., & PIATTINI, M. (2007). Towards CIM to PIM transformation: from secure business processes defined by BPMN to Use Cases. *5th International Conference On Business Process Management.* Brisbane, Australia.

SHNEIDERMAN, B. (2002). *Leonardo's Laptop: Human Needs and the New Computing Technologies.* London, England: MIT Press.

SILVER, B. (2008a). BPMS Watch: Bringing Method To The Madness. Resource Document. *BPMInstitute.org*. http://www.bpminstitute.org/articles/article/article/bpms-watch-bringing-method-to-the-madness.html. Accessed 10 November 2008.

SILVER, B. (2008b). What's Wrong With This Picture, Redux. Resource Document. *Bruce Silver's blog on business process management*. http://www.brsilver.com/wordpress/2006/11/29/whats-wrong-with-this-picture-redux/. Accessed 23rd January 2009.

SILVER, B. (2008c). What's Wrong With This Picture? Resource Document. *Bruce Silver's blog on business process management*. http://www.brsilver.com/wordpress/2006/09/06/whats-wrong-with-this-picture/. Accessed 25th January 2009.

SILVER, B. (2008d). What's Wrong With This Picture? Part 2. Resource Document. *Bruce Silver's blog on business process management*. http://www.brsilver.com/wordpress/2006/09/14/whats-wrong-with-this-picture-part-2/. Accessed 22nd January 2009.

SILVER, B. (2008e). What's Wrong With This Picture? Part 3. Resource Document. *Bruce Silver's blog on business process management*. http://www.brsilver.com/wordpress/2006/09/19/whats-wrong-with-this-picture-part-3/. Accessed 22nd January 2009.

SLACK, S. E. (2008). The business analyst in model-driven architection: Separating design from architecture. Resource Document. http://download.boulder.ibm.com/ibmdl/pub/software/dw/architecture/ar-bamda/ar-bamda-pdf.pdf. Accessed 29th October 2009.

SOLEY, R. M. (2006). Modelling all the way up... modelling all the way down. Resource Document. http://www.omg.org/mda/mda-webcasts.htm. Accessed 21st November 2007.

STEVENS, P., & POOLEY, R. (2000). *Using Uml: Software Engineering with Objects and Components.* Updated ed. Harlow, Essex, England: Pearson Education Limited.

THANGARAJ, S. (2004). Introduction to Model Driven Architecture, NCICB Software Development Processes, Facilitating Systems Interoperability. Resource Document. *Cancer Biomedical Informatics Grid, National Cancer Institute*. http://cabig.nci.nih.gov/workspaces/ICR/Meetings/ICR_Workspace/August_Face-to-Face_Meeting/F2F_MDA_Intro_Presentation.pdf. Accessed 2nd October 2007.

THOMAS, D. (2004). MDA: Revenge of the modelers or UML utopia? *IEEE Software*, 21 (3), 15 - 17.

UHL, A., & AMBLER, S. W. (2003). Point/Counterpoint. *IEEE Software*, 20 (5), 70-73.

VIDE. (2007). Visualise all model driven programming (VIDE) - The visual user interface. Deliverable 5.1 of the VIDE Project.

WHITE, S. A. (2004). Process Modeling Notations And Workflow Patterns. Resource Document. *BPTrends*. http://www.businessprocesstrends.com/deliver_file.cfm?fileType=publication&fileName=03-04%20WP%20Notations%20and%20Workflow%20Patterns%20-%20White.pdf. Accessed 27th August 2008.

WONG, P. Y. H., & GIBBONS, J. (2008). A Process Semantics for BPMN. *Proceedings of 10th International Conference on Formal Engineering Methods.* Kitakyushu International Conference Centre, Kitakyushu-City, Japan.