Towards the *Model Driven Organization*

Tony Clark[1], Vinay Kulkarni[2], Balbir Barn[1], Robert France[3], Ulrich Frank[4], and Dan Turk[3]

[1] Middlesex University, UK
[2] Tata Research Development and Design Centre, India
[3] Colorado State University, USA
[4] University of Duisburg-Essen, DE

Abstract. Modern organizations are faced with the need to rapidly respond to frequent changes arising from external business pressures. The effect of such continuous evolution eventually leads to sub-optimal configurations of underlying systems that can significantly reduce an organization's ability to provide mission-critical or highly competitive services. There has been little attempt to apply model driven principles to addressing these issues. We present a vision of a *Model Driven Organisation* (MDO) that has the potential to increase productivity by promoting integration of business processes and collaborations across the organisation whilst supporting safe and convenient adaptations that maintain system integrity. The approach is based on the use of modelling languages and simulation technologies that provide usable abstractions for understanding business contexts and goals, through to specifying IT systems, and ultimately to adapting deployed systems. The paper motivates the problem, illustrates the vision through a demonstration case, and concludes with an MDO research roadmap.

Keywords: Enterprise Architecture, Enterprise Modelling, Simulation.

1 Enterprise Systems: Problems and Challenges

Organizations such as banks and public sector institutions can be thought of as being structured in three layers [1]: The *strategic* layer defines what an organization must achieve in terms of its high-level *goals* [2], the *tactical* layer defines how an organization plans to behave and thereby achieve its goals, and the *operational* layer defines the day-to-day running of the organization in a manner that is consistent with the organization's plans. The operational layer of a modern organization is implemented in terms of a collection of inter-connected IT systems that form an *organizational platform*. An organization seeks to *align* its high-level goals with its platform so that its strategy is properly supported by its IT infrastructure [3–5]. Expressing and achieving alignment remains a key challenge for modern organizations. From a modeling perspective, alignment can be viewed as a refinement or realization relationship between models of strategic goals and IT platforms.

Alignment is important to an organization for a number of reasons. A key objective is to establish that an organization is operating *correctly*, where the notion of correctness is defined in terms of its business goals. Other uses of alignment include supporting

acquisition and *merger*, where the acquiring organization wishes to determine the similarities and differences with respect to the acquired organization in order to achieve efficiencies. Alignment can also be used to support *outsourcing*, where the goals of a service provider can be compared to a sub-set of those of the customer organization leading to the definition of service level agreements (SLAs).

Achieving goal-platform alignment is compromised by a number of issues facing a modern organization. The context of a modern organization is increasingly global and includes features such as competitors, regulatory compliance, business opportunities, threats, and unforeseen events [6–8], all of which are difficult to accommodate in a fixed structure of business goals. In addition, complex inter-dependent goals that serve multiple stakeholders must be analysed to ensure that they are not contradictory. Responding to changes in the external context, *e.g.*, to seize new opportunities or to combat external threats, requires changes to the business goals resulting in potentially large changes in the structure and behaviour of an organization.

The operational platform of an organization consists of IT components implemented using specific technologies. Changes in the platform can be imposed by the technology supplier or required by the organization in order to improve efficiency or quality. Such changes require that business alignment is re-established each time.

The scale of modern organizations is also a barrier to achieving alignment since it leads to highly complex, dynamic, inter-connected and evolving structures that can often only be characterized in terms of emergent behaviour. The resulting uncertainty about the state of an organization makes it difficult to acquire knowledge about its current state, and to construct plans that can rely on achieving a desirable future state.

In order to improve operational efficiency, agility and resilience, an organization needs to be able to define, analyse and dynamically maintain its goals, structures, resources and processes throughout its life-cycle, and to maintain their relationship (alignment) to the underlying IT platform. Current approaches to Enterprise Modelling (EM) rely heavily on human business expertise and therefore exhibit a high degree of latency in supporting key objectives such as alignment. We propose that Model Driven Engineering (MDE) can play a key role in solving this problem, but the application of MDE to EM is currently patchy at best.

Our proposed solution is a vision for the *Model Driven Organization* (MDO), where the different layers of an organization are modelled, analysed, and can be translated to a model of the underlying IT platform, whilst at the same time being accessible to all organizational stakeholders. This can be viewed as generalizing the notion of Model Driven Architecture (MDA) to the level of organizations, where the platform independent model (PIM) contains features from the strategic and tactical layers and the platform specific model (PSM) is the IT platform that runs the organization.

The paper is organized as follows: Section 2 reviews current approaches to EM and MDE and lists a number of Enterprise Architecture (EA) use-cases; Section 3 defines the vision for MDO and outlines how the EA use-cases can be addressed by a MDO; Section 4 presents an illustrative instance of the MDO vision; Section 5 concludes by presenting a research roadmap for achieving the vision.

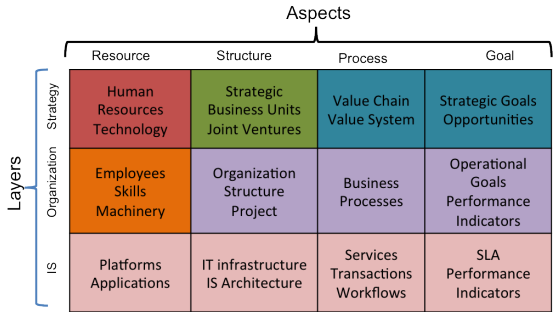


Fig. 1: Enterprise Layers

2 Current Practice

The MDO mandates advances in a range of technologies and approaches. Perhaps of most relevance is the role of EA and technologies related to MDE. This section provides a representative overview of the current state of the art and the typical use cases that illustrate how such frameworks are used in an enterprise setting.

2.1 Enterprise Frameworks

Currently, efforts in developing enterprise models for an organisation make use of established frameworks for describing key concepts and thereby contributing towards an ontology or common vocabulary. Such a framework typically presents a collection of domains focusing on describing a particular business area and could be identified as originating from frameworks such as the Zachman Framework [9]. More recently Jonkers *et al.* and Frank [1, 38] introduced frameworks. Figure 1 shows a primary decomposition of a modern enterprise using *aspects* and *layers* to help describe an organization in terms of its constituent elements as a precursor to EA modelling. Three key layers are shown: the *strategy* layer describes what an organization is trying to achieve, *i.e.*, *why* it exists; the *business* layer describes, in high-level terms the processes and resources used to achieve the desired outcomes, *i.e.*, *what* the business is doing; the *information systems* (IS) layer describes the configuration of systems used to run the business, *i.e.*, *how* the business is running. Together, these layers provide the specification and implementation of an organization.

Each layer can have multiple aspects that provide a restricted perspective of a layer. The *resource* aspect identifies the different types of resources that are relevant at each layer. The *structure* aspect provides ontologies for describing various structuring mechanisms such as roles and projects at the business layer and also subsumes structures related to information requirements. The *process* aspect describes various behavioural features at different levels of abstraction ranging from value chain models at the strategic layer through to transactions at the IS layer. The *goals* aspect focuses on intentional aspects, again at different levels of abstraction.

Such layers and aspects provide a simplified abstraction of an organization. In reality, as Jonkers et al note: ‘It is impossible and undesirable to define a strict boundary between aspects and layers because concepts that link the different aspects and layers

play one of the most important roles in a coherent architectural description.' In one sense, Figure 1 merely offers a framework for a reference description, albeit one that can be further detailed, as in the case of the Archimate Modelling Language described by Jonkers at al (ibid), or, in our case, one that can be the basis of a reference model for an organisation to support our notion of the MDO.

EA frameworks such as that outlined above are numerous and complex because of the nature of the problem they are trying to address. While the Zachman Framework is perhaps the originator, others include: the Reference Model for Open Distributed Processing (RM-ODP) [10, 11]; Open Group's framework TOGAF [12] and related frameworks for the Department of Defense (DODAF [13]), Federal processing (FEAF), UK Ministry of Defence (MODAF) [14]. Their general tendency is to add features and descriptive capability but result in bloated, hard to manage and essentially diagrammatic approaches.

2.2 Enterprise Modelling and Analysis

Enterprise Modelling aims to capture the essentials of a business, its IT and its evolution, and to support analysis of this information using a coherent whole of principles, methods, and models in the design and realisation of an enterprise's organizational structure, business processes, information systems and infrastructure [15]. Examples of analysis possible using EA includes: strategic planning, process optimisation, alignment between business functions and IT systems and business change for describing the current state of a business (as-is) and a desired state of a business (to-be). [16–20, 3]: As noted in section 1, of particular interest to the MDO is the historically thorny but important issue of business and IT alignment. This was first noted in 1977 by Mclean [21] but remains prevalent, and EA approaches have been used to address this issue [1]. Several methods and languages have emerged to provide 'whole' methods for EA, partly to address coherence issues across large EA Frameworks as described above. Example of 'whole' methods are MEMO [22]. Pereira and Sousa [11] also introduced a method that is overlayed on the Zachman framework.

In general, modelling languages for expressing features of an EA, such as ArchiMate [7], TOGAF and SysML, are often very broad, that is, they provide a wide range of features for expressing concepts familiar to a business analyst, but they lack the rigour needed to support and automate aspects of the use cases that will be described later in this section.

Further, as the scope of EA has extended to the upper layers of the conceptual framework shown in Figure 1 to address intentional modelling, several approaches have useful contributions to make towards the MDO vision. Efforts to standardise intentional modelling aspects have been consolidated in the OMG Business Motivational Model (BMM) [6]. The foundational work for BMM can be traced back to goal oriented requirements engineering (GORE) techniques [23] such as i* [24, 25] and KAOS [26, 27, 8]. Quartel et al [2] propose a language called ARMOR which provides an intentional modelling capability to the Archimate language that relies on the limited semantics provided by Archimate.

Approaches that support modeling different views of a system are beginning to appear in the software engineering arena [28]. These approaches utilize meta-models and

domain specific languages. These approaches are not yet supported by mainstream EM technologies. Similarly languages and tools for EM such as ARIS [29] or MEMO [22] focus on representing a company from different perspectives to support various kinds of analysis. A key issue with such modelling tools is that they are not integrated with enterprise systems. In contrast, Frank and Strecker [30] describe an approach to integrate enterprise models with enterprise systems that they call *self-referential enterprise systems* but the proposed technologies suffer from the limitations of main-stream programming languages.

2.3 Architectural Styles

An EA can be organised in a variety of ways, but most involve the identification of logical or physical business units, or components, that manage their own data and resources, implement a collection of business processes, and communicate with other components using a variety of message passing styles. A Service Oriented Architecture (SOA) involves the publication of logically coherent groups of business functionality as interfaces, that can be used by components using synchronous or asynchronous messaging [31]. An alternative style, argued as reducing coupling between components and thereby increasing the scope for component reuse, is Event Driven Architecture (EDA), whereby components are event generators and consumers. EDA is arguably more realistic in a sophisticated, dynamic, modern business environment, and can be viewed as a specialization of SOA where communication between components is performed with respect to a single generic event interface [32, 33].

2.4 Model Driven Engineering

While MDE is broad in scope, we focus on those aspects that address the issues pertinent to the MDO vision. Many of the aspects and the associated challenges are described in the MDE roadmap paper by France and Rumpe [34]. In particular, technology supporting integrated use of multiple general-purpose and domain specific modeling languages (DSMLs) [35] may be one of the key enablers of MDO. In an MDO, use of models expressed in different languages are inevitable, simply because of the variety of roles that models will play in operating and evolving an organization. Furthermore, enterprise aspects addressed by models will span different enterprise layer and levels of abstraction, and thus technologies supporting model manipulations (e.g., model authoring, versioning, transformation and composition) will also be critical to successful realization of an MDO.

Emerging work on using models as the primary means to managing and adapting systems at runtime (referred to as models@runtime) [36], are also applicable to MDOs. In an MDO, models will be used operate and evolve an organization. Stakeholders (e.g., employees, business partners, vendors, customers) will access services provided by an MDO by manipulating models. In addition, enterprise architects will have the capability of evolving an enterprise through models that are, in a sense, causally connected to underlying systems at the operational layer. This raises the models@runtime concept to the organizational layer.

2.5 Use-Cases for Enterprise Architecture Analysis

Enterprise Architectures are built to support *use-cases* related to managing and evolving an organization. For example, *directive development* is concerned with developing directives that express *how* a business operates; *business intelligence* describes how a CEO is informed of the state of the organization at any level; *resource planning* involves the allocation of business resources to processes; *impact analysis* covers a variety of analyses used to measure the effect a proposed change has on an organization; *change management* involves describing the context and requirements for changes in any aspect of the business, including the construction of *as-is* and *to-be* analysis and the calculation of the ROI for any proposed change; *regulatory compliance checking* establishes that an organization meets some externally imposed constraints on its operating procedures; *risk analysis* identifies dangers, both internal and external, that can affect the successful operation of the organization; *acquisition and merger* involves the comparison of two organizations to identify their similarities and differences with respect to achieving a goal; *outsourcing* involves the identification of services that can be supplied by an external partner.

Supporting the above and other EA use cases is challenging. For example, enterprise architects need to ensure that the models accurately describe relevant aspects of an organization at an abstraction level that supports specific purposes (e.g. the use cases above). A significant challenge relates to supporting multiple perspectives. Support for the separation of concerns and division of labour principles is required to deal with system complexity and to achieve economies of scale. The accompanying demands for different enterprise perspectives that are associated with specialized terminologies and processes point to the need for multiple models, possibly written in a variety of modeling languages. In order to foster collaboration across different perspectives, the various models of an enterprise should be integrated through common concepts.

Evolving an enterprise system using current approaches is also challenging. Incomplete information about the current state of an organization, imprecise understanding of the impact of a proposed change, and time-, effort- and cost-intensive introduction of change are the principal reasons why only a few transformative projects get completed within budget, and even fewer deliver the desired ROI (Return On Investment)[5]. Moreover, the need for the enterprise to remain operational while transformations are effected adds further complexity.

3 A Vision for the Model Driven Organization

The previous sections outlined the problems facing modern organizations and motivated the use of modelling as the basis for a solution. MDA seeks to solve many of the problems associated with the development of single IT systems using model-based techniques. We seek to establish an approach that extends the principles of MDA to an organization in which models are the primary means for interacting with and evolving the systems that drive the organization. We will refer to this approach as the Model

[5] Metrics for Enterprise Transformation `http://tinyurl.com/d83ctvd`

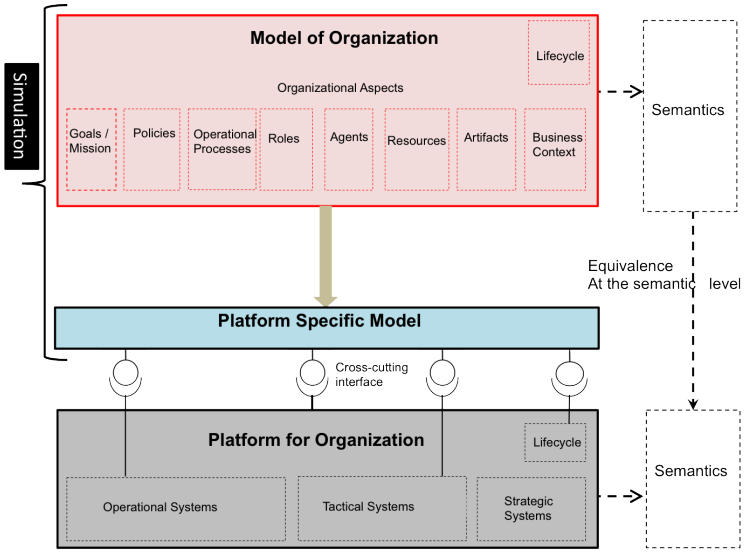


Fig. 2: The Essential Characteristics for a Model Driven Organization

Driven Organization (MDO). This section provides a definition for MDO and discusses its key features.

> **def:** A *Model Driven Organization* uses models in the analysis, design, simulation, delivery, operation, and maintenance of systems to address its strategic, tactical and operational needs and its relation to the wider environment.

Figure 2 shows the characteristics of an MDO consisting of a model of the organization (*Model of Organization*), a model of the platform that runs the organization(*Platform Specific Model*), and a transformation between them, as described below:

Model of the Organization: An organization consists of a collection of interacting aspects ranging from goals and missions through to the business context. Figure 2 shows a non-exhaustive collection of aspects. Each aspect is supported by a domain-specific language whose models provide a basis for simulation, *what-if*, and *if-what* analysis. A key challenge here is to find an integration mechanism for these languages that also supports an organization's life-cycle as it responds to changes in the business context.

Platform for Organization: An organization uses a collection of IT systems to realize its business functions. A platform consists of domain specific applications, software infrastructure, network infrastructure, hardware infrastructure. We make a distinction between three types of IT systems *strategic*, *tactical*, and *operational systems* that correspond to the different needs within the organization [37]. For example, a balanced score card system is a *strategic* system, and payroll is an *operational* system. A platform provides a collection of interfaces that can be used to drive its IT systems. Cross-cutting concerns that involve multiple IT systems are supported by appropriate interfaces. The life-cycle of the platform involves upgrades, interactions with external systems, configurations, and is supported by a dedicated interface.

Platform Specific Model: The PSM is a model that is used to express configurations of the IT systems supported by the platform described above. A PSM is derived from a Model of the Organization by a semantics preserving transformation. This requires that both the organizational modelling language and the PSM language have well-defined semantics. Our proposal is that realizing the MDO vision is beneficial for an organization as it becomes possible to perform *what*, *why* and *how* analysis on its structure and behaviour, where this would otherwise be very difficult or impossible. Making the dependencies explicit has at least two benefits: *system integrity* is improved because the model allows the effects of change to be propagated throughout the organization; *transparency* is improved due to a reduction in the complexity of the organization through the use of domain specific models. Note that in both cases we talk of improvements because there will always be parts of a system that rely on human control and expertise, and which cannot be modelled. Since dependencies are made explicit, it is possible to precisely measure the effect of a given change and also to provide organizational views from different perspectives. Consider the enterprise use-cases outlined in section 2. Our vision for the MDO must be seen to support these use-cases in a way that increases organizational effectiveness. In all cases domain-specific modelling will be used to provide languages that support the various layers and aspects of an organization. Below we revisit each use-case in turn and discuss how MDE can help:

Directive Development: Directives place constraints on how a business operates and as such acts as domain specific invariants on business processes. Expressing both business processes and associated directives as invariant models allows the directives to be continuously applied to the processes as they are enacted. Domain specificity allows directive violations to be reported to the appropriate stakeholders.

Business Intelligence: Business intelligence can be achieved by expressing both the state and objectives of a business as models. Since the definition of an *objective* is that it must be measurable then it will be possible to use the state model to provide all management stakeholders with real-time views of current progress against objectives (and therefore overall goals).

Resource Planning: Plans can be constructed by comparing goal models with models of resources and of business processes. The dependencies between these models will allow changes to the aims, processes or internal organization of the business to be propagated. For example, the impact of the ability of an organization to achieve its goals can be determined after a reduction in a specific type of resource.

Impact Analysis: This requires modelling dependencies between elements in an organization. We envisage a situation where all aspects of an organization are represented in a model and therefore changes to any aspect or level can be propagated throughout.

Change Management: Proposed changes are analysed by constructing a model of a business *as-is* and *to-be*. Since our approach is to model all aspects of an organization, it is possible to precisely compare the two models and to establish that measures such as KPIs are maintained or improved by the proposed change. Model transformation techniques can be used to define organizational change. Furthermore, the models can be used as the basis for checking or even automatically constructing a change plan model.

Regulatory Compliance: This is achieved by providing the regulatory body with evidence that required processes are being implemented. If an organization is run from models then this is easily achieved by auditing the models. Furthermore, if regulations are published as models, including descriptions of valid evidential compliance, then it would be possible to upload the regulation model and for an organization to be automatically configured to provide the required evidence.

Risk Analysis: Analysis of risk can be achieved using models in a number of ways. Internal risks impact the ability of a business to achieve its goals and therefore analysis of models provides a way to both statically and dynamically quantify risks. For example, the reliability profile for an IT component can be used as part of a simulation to determine the probability of a given business goal failing. External risks are more difficult to quantify, however intentional models can be used to attribute probabilities to external events acting as obstacles for organizational goals. For example, the likelihood of a key customer moving to a competitor.

Acquisition and Merger: This is a special case of business change where one organization assimilates another. Modelling can play a key role here by supporting the comparison of the two organizations and determining similarities and differences. Domain specific model comparison can be used to automatically determine which processes of an acquired business are already performed by the acquiring business and to compare the efficiency of both. Model merge techniques can be used to compare different possible outcomes of an acquisition. Modelling can also help support speculative acquisition by comparing the goal models of two companies.

Outsourcing: This provides an opportunity for using model transformation and model slicing techniques. Given a model of an organization and a service provider it will be possible to isolate that part of an organization to be outsourced and then to transform the organization by slicing. Models of service level agreements can be used to automatically check required levels of provision.

Organizational modelling provides opportunities for standardization through frameworks and languages. In turn, repositories of good practice can be established and possibly accredited so that quality levels of organizational behaviour can be defined.

To achieve the vision, an organization will be represented by a set of integrated models as described in [38], each of which supports a specific perspective of an enterprise and associated tools [30]. Depending on preferences and skills, the models can be represented using, for example, graphical diagrams, text, tables, and cover different levels of abstraction from the instance-level to meta-levels. Thus, model-centric systems provide users with versatile tools to navigate, analyze, modify and interact with the organization and with other stakeholders that have different perspectives.

4 An Illustration of the Model Driven Organization

Figure 2 shows an overview of the features of an MDO. The general structure can be specialized to a domain by limiting the operational aspects and addressing a specific class of platforms. In practice, it is likely there there will be many different MDO instances that target different domains. This section provides an overview of such an

instance in terms of a requirements for an MDO IT Plant followed by a description of the key features that such an MDO might contain.

4.1 Example MDO

Organizations use IT systems as a basis for their strategic, tactical and operational requirements. We will refer to the systems collectively as an *IT Plant* (ITP). The costs associated with these systems are categorized as either transactional (*run the business*) or transformational (*change the business*). Reducing such costs are a significant issue for any organization. Outsourcing may be used to bring down the costs by transferring development and maintenance of IT systems to low cost geographies. Other approaches involve consolidation and rationalization of hardware infrastructure, harmonization of technology infrastructure.

Outsourcing and hardware consolidation are fast approaching the point of diminishing (if not zero) return and harmonization of software infrastructure can bring only so much benefit. Individual systems within a traditional ITP are typically associated with specific functional requirements. Therefore any amount to improvement to an individual IT system is unlikely to guarantee improvement in the ITP as a whole. Thus, the current practice seems to be approaching its limits in terms of cost effectiveness.

Consider a service provider who wishes to supply a domain-specific ITP. The provider will want to cater to the IT needs of multiple organizations through a single multi-tenant ITP using a flexible and low-cost configuration mechanism. Each customer must be able to easily determine whether the service provider can meet their functional and non-functional requirements. Conversely, the service provider must be able to easily demonstrate that they meet the requirements of each customer to the IT services it manages without duplicating the ITP for each new customer. Such a service provider represents a new business model that enables servicing of transactional and transformational IT needs in outcome-based pricing and on operational risk sharing basis. Clearly a win-win situation for both organizations and ITP providers.

The MDO framework shown in figure 2 can be specialised to support the ITP outlined above as follows:

Model of the Organization: An organization will specify their IT needs in terms of models including descriptions of processes, services, data, user experience, NFC, SLA, pricing and risk. Current EA practice advocates use of a subset of these models but only as blue prints that need to be interpreted by a human expert. On the contrary, an MDO supports analysis and simulation for functional and non-functional properties. Variability will be explicit in these models wherever required. Thus, the organization model (PIM) can be used as the basis for a commercial agreement between the organization and the service provider.

Platform for the Organization: The platform in this case is the ITP providing domain specific interfaces that can be used to configure and run IT applications.

Platform Specific Model: The service provider will use a domain-specific language to express the features of the IT applications run on the platform. The language will support IT-level concepts such as processes, workflows, test-cases, and services. The language will support analysis and simulation so that the provider can supply the customer with concrete evidence that the required services can be provided and meet defined

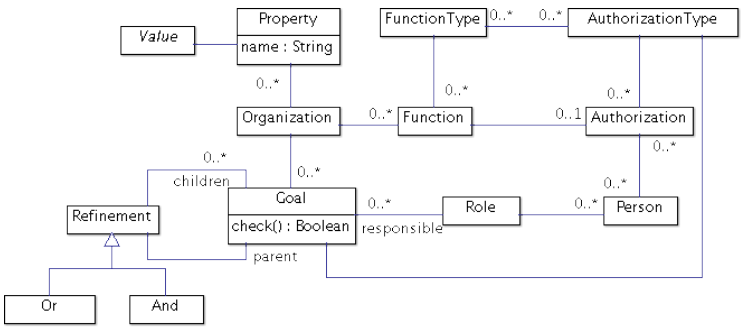


Fig. 3: An example Organization Model

quality criteria. The PSM language will use product-line and variation-point techniques to ensure that both inter- and intra-customer variation requirements can be met. The PSM is used to control the platform; this can be through a variety of techniques such as code generation, executable models and configuration of tables that control choices on the platform.

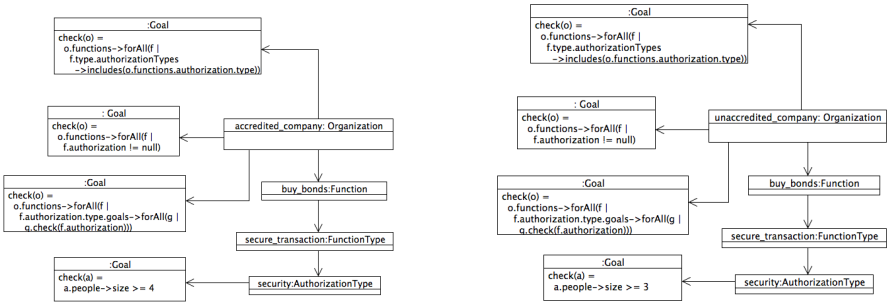
The MDO ITP process involves the construction of an organizational model, possibly containing variation points. Such a model will be constructed by the customer, most likely assisted by a consultant. The service provider will use the analytical and simulation properties of the PIM language to supply a cost to the customer. This is achieved through the use of a semantics preserving transformation from the PIM to the PSM, effectively *compiling* the PIM into a form that can be used to run on the platform, thereby significantly reducing the cost to the service provider. The transformation will be performed by a service provider expert, possibly assisted by a *knowledge based system* that is used to manage expertise in matching organizational requirements to the ITP. The rest of this section provides a simple example of such an MDO ITP.

4.2 Model of the Organization

Figure 3 shows a simple model that could be used to capture part of the IT needs of an organization. Each function corresponds to a required service. In this particular class of organization, each function requires some authorization and there are a number of function types and authorization types. For example, if the organization is a bank then functions might be grouped in terms of accounts, shares or bonds, and the authorization might be grouped in terms of security levels.

An organization has a collection of goals that define what it is trying to achieve and also define what must hold in order for authorization to be valid. The goals are structured in and/or trees. We will use this feature to represent a variation point in the required service provision.

As an example, consider the Banking industry. A Bank is an *Organization* that has *Goals*. Some goals that Banks have are *Make Money for Shareholders*, *Comply with Regulations*, and *Provide Security and Privacy for Customers and Transactions*, among others. Examples of *Functions* that Banks have are *Manage Deposits*, *Manage Automatic Transactions*, *Process Inter-bank Transactions*, and *Issue Bonds*. Banks also have



(a) Accredited Organization (b) Non-Accredited Organization

Fig. 4: Variations on a bond purchase

various *Authorizations* performed for these *Functions*. Some of these might include *Security Authorization* and *Regulatory Oversight Authorization*. Much more detail would be desired in a full model of Banking, but this illustrates how the meta-model in Figure 3 could be instantiated into a model of a real-world organization.

As an example of how models can be used in a MDO, consider a scenario in which a bank requires a *bond purchase* service from a provider who implements an ITP. Such an operation requires authorization from several agents within the organization. Normally such authorization requires 3 people; this is called *6-eye* authorization. However, if the bank has been accredited by the FSA then the regulation can be satisfied with 2 people: *4-eye* authorization.

Figure 4 shows the organizational models for two different organizations. The first is an accredited bank and the goal associated with the authorization type requires that the number of people associated with each authorization for the bond buying function is a minimum of 4. The second is a non-accredited bank, therefore the goal requires 3 people.

Since the requirements for the bond buying service are explicitly expressed as part of the organizational model, the service provider can configure the ITP to give an appropriate level of checking to each different customer. This might take the form of sending 2 or 3 secure emails to people in the appropriate roles and waiting to receive replies containing secure sign-off.

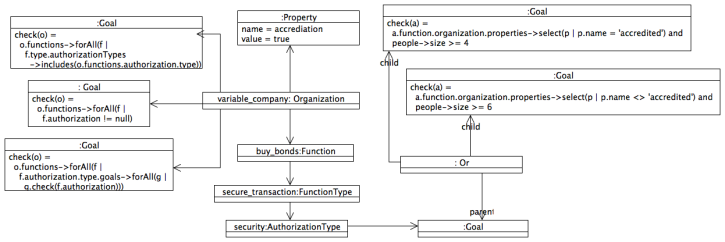
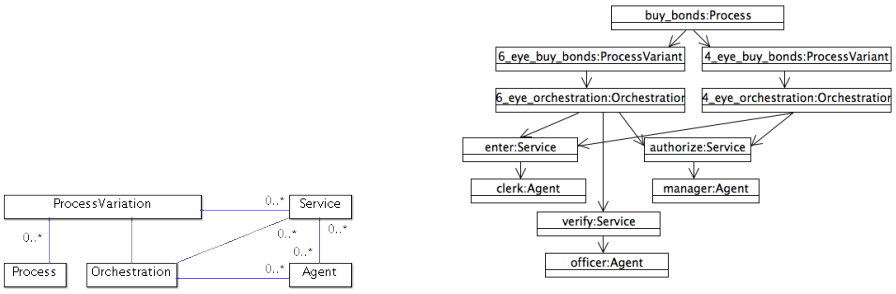


Fig. 5: Organization with variation



(a) Platform Specific Model (b) Input to Platform

Fig. 6: Platform specific bond purchasing

The ITP might use variation in its implementation to service both customers in figure 4, however the different types of organization do not require any variability in their provision. Consider the case of a company that currently is not accredited but expects to achieve FSA accreditation in the near future. Their bond buying service will need 6-eye authorization initially, but will want to change to 4-eye if they achieve accreditation. Furthermore they do not want to pay extra for this change of service because they will inform the ITP in advance.

Figure 5 shows an organizational model for such a company. The variation point is achieved by including a boolean property called *accreditation* that is used in the goal of the authorization type. The ITP can take account of such variations in order to pre-configure the service. Since models are used to express the requirement and to configure the ITP, it is possible for the service provider to manage the cost of providing the variability by generating the service variations from the organizational model.

4.3 Platform Specific Models

A customer provides an organizational model of their IT requirements which is transformed into a platform specific model used to configure the service provider's ITP. Figure 6(a) shows a simple model of processes that could use used as the target of such a transformation. Processes are used to realize the functions required by customers, a process may have a number of pre-defined variants each of which is implemented using an orchestration of services enacted by agents. The model is very simple and achieves variability through pre-defined process variations. In practice we envisage sophisticated methods from the field of product-line engineering to be applied in order to achieve the maximum static and dynamic flexibility.

Figure 6(b) shows an instance of the platform specific model that corresponds to the transformation of the organizational model shown in figure 5 which, in turn, subsumes the organizational models shown in figure 4.

5 A Research Roadmap for the Model Driven Organization

Realizing the MDO vision described in section 3 requires input from many research fields including Enterprise Modelling, Enterprise Architecture and Model Based Engi-

neering. In practice, we envisage a situation where there may be a large number of MDO categories each of which is specialized to a particular domain and which requires input from specific sub-fields. The MDO requirements are reviewed with respect to the state of the art in section 2 and performs a gap-analysis in order to speculate on a possible MDO research roadmap.

Model of Organization: Features of an organization such as goals, processes, organizational structure, services, data, risk, value, *etc.*, and inter-relationships between them need to externalized. Work on these aspects is typically reported independently, for example [39–42][6,7] [8]. There is little work reported on modeling the inter-relationships between these models. Individual models (especially [40–42]) require human experts for interpretation and lack a precise semantics (effectively being 'correct by definition') necessary to support analysis and semantics preserving transformations.

Analysis and Simulation: There is a need to establish functional and extra-functional properties of an organizational model in qualitative and/or quantitative terms. Due to the inherent uncertainty in the domain, fuzzy or probabilistic techniques may help address qualitative analysis [43]. Analysis techniques exist for individual aspects of an organization, but method support and technology to combine the results of individual analysis is lacking. At present it is not possible to simulate all aspects of an organization. Operational system models can be constructed for certain aspects of an organization in terms of a small set of primitives and powerful simulation machinery [44] and it may be possible to extend this approach to other aspects of an organization by including features such as planning [45] and model checking.

Contract Specification: The relationship between the PIM and the PSM can be viewed as a contract between the organizational needs and the platform services in a particular domain. To be effective the relationship needs to utilize model transformation techniques. However, most current transformation techniques have been developed to address software development concerns and are weak in terms of verification. Model transformations for MDO will need to transform constraints (*e.g.*, SLAs), architectures, process descriptions, non-functional properties, *etc.* Given the complexity of the source model, such a transformation will be large and therefore MDO refinement techniques (possibly aided using a KBS) might be appropriate.

Platform for Organization: The organizational platform must be modelled in sufficient detail so that an implementation of a configurable extensible platform can be derived and used (under human supervision) to monitor, evolve and adapt the organization. A method, either manual or partially automated is needed to establish verification through traceability between the contract specification (PIM) and platform specification (PSM). Modelling has been shown to support single IT systems in terms of user interface, data and data access, on-line and batch functionality, reports, *etc.*, to support design-time and run-time configuration of a single IT system [46], and to generate efficient implementations [47]. It is also possible to specify interactions between

[6] Business Process Model Notation (BPMN), v. 2.0, 2011. OMG: `www.omg.org/spec/BPMN/2.0`

[7] The Web Service Modeling Language WSML `http://www.wsmo.org/TR/d15386607/d16.1/v0.3/20070209/d16.1v0.3_20070209.pdf`

[8] UML 2.0 Superstructure Specification.OMG, Needham (2004)

applications as an orchestration or choreography [6]. However, little work is reported on application architecture to support unforeseen extensibility. The adaptation concept needs to be extended individually to every constituent such as business processes, services, databases, user interfaces, *etc.*, and collectively to the whole platform. This would involve building further on the ideas of software product lines [48] and architecture description languages [49].

Testing the Platform: At present it is possible to specify application behavior and to generate test cases and test data for coverage related assurance [50]. Emerging work described product-line testing for a set of applications that exhibit high commonality and well-defined variability [51]. Automation harnesses for regression testing have been around for years, however, incremental *i.e.,* change-specific testing is still a problem. Moreover, these ideas need to be extended to cover the whole platform. Another, and probably more important, problem is to establish testability of the platform.

Deploying the Platform: It is unlikely that an organization will run entirely as an MDO. Partial migration to an MDO leads to dependency issues between the platform and the non-platform IT systems and will require modification or decommissioning. The identification of such dependencies may require analysis of existing systems, probably in terms of their execution logs. These activities need to be automated and verified where possible.

Domain Models: Realising the MDO will require input from many different stakeholders and domains of expertise. Many of the modelling techniques needed to implement an MDO will cut across domains such as banking, insurance, telecom, *etc.* This will require advances in domain engineering, ontologies, meta-modelling, and domain-specific language engineering in order to achieve the level of integration required.

6 Conclusion

This exploratory paper describes the problems that occur when modern organizations seek to achieve strategic alignment of business goals with IT systems and to support EA use-cases. Our proposal is to move towards a Model Driven Organization whereby Model Based Engineering techniques are used to allow stakeholders to specify, analyse and interact with an organization through the use of platform independent models that are translated into technology specific models suitable for deployment on an organization platform. Our vision generalizes the notion of MDA so that it can be applied at the enterprise level and thereby address alignment problems. We have provided a research roadmap that indicates where research effort is required in order to achieve the vision.

References

1. H. Jonkers, M. Lankhorst, R. Van Buuren, S. Hoppenbrouwers, M. Bonsangue, and L. Van Der Torre, "Concepts for modeling enterprise architectures," *International Journal of Cooperative Information Systems*, vol. 13, no. 3, pp. 257–287, 2004.
2. D. Quartel, W. Engelsman, H. Jonkers, and M. Van Sinderen, "A goal-oriented requirements modelling language for enterprise architecture," in *Enterprise Distributed Object Computing Conference, 2009. EDOC'09. IEEE International*. Ieee, 2009, pp. 3–13.
3. Y. Chan and B. Reich, "IT alignment: what have we learned?" *Journal of Information Technology*, vol. 22, no. 4, pp. 297–315, 2007.

4. J. Henderson and N. Venkatraman, "Strategic alignment: Leveraging information technology for transforming organizations," *IBM Systems Journal*, vol. 32, no. 1, pp. 4–16, 1993.
5. P. P. Tallon and K. L. Kraemer, "Investigating the relationship between strategic alignment and it business value: the discovery of a paradox," *Creating Business Value with Information Technology: Challenges and Solutions. Hershey, PA: Idea Group Publishing*, pp. 1–22, 2003.
6. B. Berkem, "From the Business Motivation Model (BMM) to Service Oriented Architecture (SOA)," *Journal of Object Technology*, vol. 7, no. 8, 2008.
7. M. M. Lankhorst, H. A. Proper, and H. Jonkers, "The anatomy of the archimate language," *IJISMD*, vol. 1, no. 1, pp. 1–32, 2010.
8. A. Van Lamsweerde, "Requirements engineering: from craft to discipline," in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*. ACM, 2008, pp. 238–249.
9. J. Zachman, "A framework for information systems architecture," *IBM Systems Journal*, vol. 38, no. 2/3, 1999.
10. ITU, "Basic reference model of open distributed processing - part 1: Overview and guide to use," in *ITU Recommendation X.901 — ISO/IEC 10746-1*. ISO/ITU, 1994.
11. C. Pereira and P. Sousa, "A method to define an enterprise architecture using the zachman framework," in *Proceedings of the 2004 ACM symposium on Applied computing*. ACM, 2004, pp. 1366–1371.
12. J. Spencer *et al.*, *TOGAF Enterprise Edition Version 8.1*, 2004.
13. D. Wisnosky and J. Vogel, "DoDAF Wizdom: A Practical Guide to Planning, Managing and Executing Projects to Build Enterprise Architectures Using the Department of Defense Architecture Framework (DoDAF)," 2004.
14. B. Biggs, "Ministry of defence architectural framework (modaf)," in *IEE Seminar Digests*, vol. 43, no. 2005, 2005.
15. M. Lankhorst, "Introduction to enterprise architecture," in *Enterprise Architecture at Work*, ser. The Enterprise Engineering Series. Springer Berlin Heidelberg, 2009. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-01310-2_1
16. M. Ekstedt, P. Johnson, A. Lindstrom, M. Gammelgard, E. Johansson, L. Plazaola, E. Silva, and J. Lilieskold, "Consistent enterprise software system architecture for the cio - a utility-cost based approach," in *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04)*, 2004.
17. C. Riege and S. Aier, "A Contingency Approach to Enterprise Architecture Method Engineering," in *Service-Oriented Computing–ICSOC 2008 Workshops*. Springer, 2009.
18. K. Niemann, *From enterprise architecture to IT governance: elements of effective IT management*. Vieweg+ Teubner Verlag, 2006.
19. T. Bucher, R. Fischer, S. Kurpjuweit, and R. Winter, "Analysis and application scenarios of enterprise architecture: An exploratory study," in *10th IEEE International Enterprise Distributed Object Computing Conference Workshops, 2006. EDOCW'06*, 2006.
20. L. Paape and R. Speklé, "The adoption and design of enterprise risk management practices: An empirical study," *European Accounting Review*, vol. 21, no. 3, pp. 533–564, 2012.
21. E. McLean and J. Soden, *Strategic planning for MIS*. John Wiley & Sons Inc, 1977.
22. U. Frank, "Multi-perspective enterprise modeling (memo) conceptual framework and modeling languages," in *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*. IEEE, 2002, pp. 1258–1267.
23. A. van Lamsweerde, "Goal-oriented requirements enginering: a roundtrip from research to practice [enginering read engineering]," in *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International*. IEEE, 2004, pp. 4–7.
24. E. Yu, "Towards modelling and reasoning support for early-phase requirements engineering," in *Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on*. IEEE, 1997, pp. 226–235.
25. E. Yu and J. Mylopoulos, "Understanding ẃhyín software process modelling, analysis, and design," in *Proceedings of the 16th international conference on Software engineering*. IEEE Computer Society Press, 1994, pp. 159–168.
26. A. Dardenne, A. Van Lamsweerde, and S. Fickas, "Goal-directed requirements acquisition," *Science of computer programming*, vol. 20, no. 1-2, pp. 3–50, 1993.

27. E. Letier and A. Van Lamsweerde, "Reasoning about partial goal satisfaction for requirements and design engineering," in *ACM SIGSOFT Software Engineering Notes*, vol. 29, no. 6. ACM, 2004, pp. 53–62.
28. A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke, "Viewpoints: A framework for integrating multiple perspectives in system development," *International Journal of Software Engineering and Knowledge Engineering*, vol. 2, no. 1, pp. 31–57, 1992.
29. A.-W. Scheer, O. Thomas, and O. Adam, "Process modeling using event-driven process chains," *Process-Aware Information Systems*, pp. 119–146, 2005.
30. U. Frank and S. Strecker, "Beyond ERP Systems: An Outline of Self-Referential Enterprise Systems," Institute for Computer Science and Business Information Systems (ICB), Duisburg-Essen University, Germany, ICB Research Report 31, April 2009, http://alturl.com/6kmcd.
31. D. Barry, *Web services and service-oriented architecture: the savvy manager's guide*. Morgan Kaufmann Pub, 2003.
32. L. David, "The power of events: an introduction to complex event processing in distributed enterprise systems," 2002.
33. A. Buchmann and B. Koldehofe, "Complex event processing," *it-Information Technology*, vol. 51, no. 5, pp. 241–242, 2009.
34. R. France and B. Rumpe, "Model-driven development of complex software: A research roadmap," in *2007 Future of Software Engineering*. IEEE Computer Society, 2007, pp. 37–54.
35. M. Mernik, J. Heering, and A. Sloane, "When and how to develop domain-specific languages," *ACM Computing Surveys*, vol. 37, no. 4, pp. 316–344, 2005.
36. G. Blair, N. Bencomo, and R. France, "Models@ run.time," *Computer*, vol. 42, no. 10, pp. 22–27, Oct.
37. F. W. Dewhurst, K. D. Barber, and M. C. Pritchard, "In search of a general enterprise model," *Management Decision*, vol. 40, no. 5, pp. 418–427, 2002.
38. U. Frank, "Multi-perspective enterprise modeling: foundational concepts, prospects and future research challenges," *Software and Systems Modeling*, pp. 1–22, 10.1007/s10270-012-0273-9.
39. E. Yu, M. Strohmaier, and X. Deng, "Exploring intentional modeling and analysis for enterprise architecture," in *Enterprise Distributed Object Computing Conference Workshops, 2006. EDOCW '06. 10th IEEE International*, oct. 2006, p. 32.
40. C. Abrams, J. von Kanel, S. Muller, B. Pfitzmann, and S. Ruschka-Taylor, "Optimized enterprise risk management," *IBM Systems Journal*, vol. 46, no. 2, pp. 219 –234, 2007.
41. J. Gordijn, E. Yu, and B. van der Raadt, "E-service design using i* and e/sup 3/ value modeling," *Software, IEEE*, vol. 23, no. 3, pp. 26 –33, may-june 2006.
42. R. E. Miles, C. C. Snow, A. D. Meyer, and H. J. Coleman Jr, "Organizational strategy, structure, and process," *Academy of management review*, pp. 546–562, 1978.
43. P. Klinov and B. Parsia, "Pronto: Probabilistic ontological modeling in the semantic web," in *International Semantic Web Conference (Posters & Demos)*, ser. CEUR Workshop Proceedings, C. Bizer and A. Joshi, Eds., vol. 401. CEUR-WS.org, 2008.
44. D. Wright and D. H. Meadows, *Thinking in systems: a primer*. Routledge, 2012.
45. J. A. Hendler, A. Tate, and M. Drummond, "Ai planning: Systems and techniques," *AI magazine*, vol. 11, no. 2, p. 61, 1990.
46. M. C. Huebscher and J. A. McCann, "A survey of autonomic computingdegrees, models, and applications," *ACM Comput. Surv*, vol. 40, no. 3, pp. 1–28, 2008.
47. V. Kulkarni and S. Reddy, "Model-driven development of enterprise applications," *UML Modeling Languages and Applications*, pp. 118–128, 2005.
48. P. Clements and L. Northrop, *Software product lines*. Addison-Wesley, 2002.
49. P. C. Clements, "A survey of architecture description languages," in *Proceedings of the 8th international workshop on software specification and design*. IEEE Computer Society, 1996, p. 16.
50. J. Offutt and A. Abdurazik, "Generating tests from uml specifications," *UML 99 The Unified Modeling Language*, pp. 76–76, 1999.
51. A. Reuys, E. Kamsties, K. Pohl, and S. Reis, "Model-based system testing of software product families," in *Advanced Information Systems Engineering*. Springer, 2005, pp. 379–380.