Intentional Modeling for Problem Solving in Enterprise Architecture

Sagar Sunkle, Vinay Kulkarni and Suman Roychoudhury
Tata Research Development and Design Center, Tata Consultancy Services,
54B, Industrial Estate, Hadapsar, 411013, Pune, India

Keywords: Enterprise Architecture, Intentional Modeling, Problem Solving.

Abstract: Taking and executing correct decisions is critical in enterprise systems which are characterized by rapid changes along interconnected dimensions. Enterprise architecture (EA) frameworks offer holistic treatment of enterprise systems but constitute only one part of the solution to problems arising due to organizational changes. The other, less explored part is the ability to explicate and analyze the intentions behind major decisions. We investigate a step-by-step approach where intentional modeling is treated as a problem solving technique. In our approach, an intentional model devoid of goals is obtained from the existing EA model via mapping. It is expanded by representing the problems due to organizational changes as goals and soft goals and alternative solutions to them. The final intentional model is transformed back to an actionable EA model via the same mapping. In the case study, we re-imagine the evolution of our model-driven software development unit as an enterprise where two stages in its evolution are treated as as-is and to-be states and the journey is captured in terms of intentional models. Initial explorations suggest that the mapping enables a clear path from as-is to to-be states of an EA model while preserving the reasoning behind every alternative chosen.

1 INTRODUCTION

In our experience in delivering 70+ large business-critical enterprise applications, we have found that the cost of incorrect decisions in building these systems and evolving them in the face of organizational changes is becoming prohibitively high (Kulkarni et al., 2010). One key reason that we witnessed as to why decisions are so hard to see through in enterprises is that a holistic view is missed. The second reason is that even if such a view is taken, the reasons behind decisions are scarcely captured or utilized to direct the change.

For an enterprise, the holistic view may be provided by enterprise architecture (EA). A number of EA frameworks and modeling languages are now routinely used by enterprises to improve business efficiency. Yet, reasons behind EA construction are hard to trace or examine and so is responding to change (Yu et al., 2006). To circumvent this, some EA methods and independent standards such as business motivation model (BMM) (OMG, 2010) provide additional modeling constructs in the form of *motivation* extension(s). These are good for highly abstracted views at the management level, but when it comes down to ground level implementations these constructs act only as blueprints rather than techniques to capture and use ***whys*** in constructing and evolving ***what*s** and ***how*s** of enterprises.

We approach this situation from the perspective of problem solving. We posit that while EAs capture details of a given change context, a mapping from EA concepts to intentional modeling concepts (Yu et al., 2006) can be used so that eminent problems are represented as (soft) goals to be reached. What-if analyses provided by intentional modeling (Horkoff and Yu, 2009) can then be used solve these problems which includes exploring alternatives from the point-of-view of involved actors/roles.

As a running case study example, we re-imagine our model-driven software development unit as an enterprise. We capture in retrospect the organizational changes that we observed over the years as problems to be solved and model-based solutions as the answers. Our specific contribution is the mapping of core elements of EA and intentional metamodels used to translate goals from and to EA models. Once various questions pertaining to the problems have been answered, the results of these analyses, in terms of addition of new actors/roles, change of responsibilities of existing actors/roles, and so on, are transferred back to EA models via the same mapping preserving traceability from ***whys*** to ***what*s** and ***how*s**.

The rest of the paper is organized as follows. Sec-

Sunkle S., Kulkarni V. and Roychoudhury S..
Intentional Modeling for Problem Solving in Enterprise Architecture.
DOI: 10.5220/0004435502670274
In *Proceedings of the 15th International Conference on Enterprise Information Systems* (ICEIS-2013), pages 267-274
ISBN: 978-989-8565-61-7
Copyright © 2013 SCITEPRESS (Science and Technology Publications, Lda.)

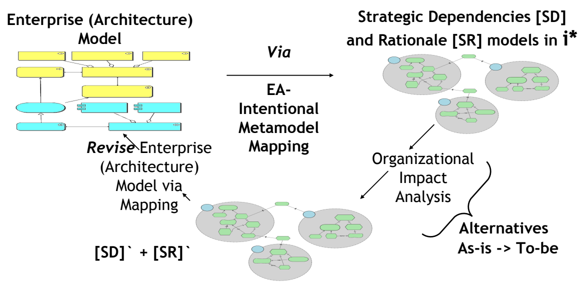
Enterprise (Architecture) Model
Via
EA-Intentional Metamodel Mapping
Strategic Dependencies [SD] and Rationale [SR] models in i*
Organizational Impact Analysis
Alternatives As-is -> To-be
Revise Enterprise (Architecture) Model via Mapping
[SD]` + [SR]`

Figure 1: Solving problems in enterprise architecture with i* intentional modeling.

tion 2 describes the mapping between generic EA and intentional metamodels and introduces the case study. Section 3 captures in terms of intentional models, the problems faced by our model-driven software development unit and presents alternative solutions. Section 4 describes how the results of analyses over the intentional models are transferred back to EA models with which to take action. Section 5 reviews related work and Section 6 concludes the paper.

2 USING INTENTIONAL MODELS

While EA frameworks capture the working of the enterprise or the ***whats*** and the ***hows***, the treatment of ***whys***, even with the extended constructs for *motivation* in them, is of *blueprint* nature (Wagter et al., 2012). We believe that in order to capture the reasons behind enterprise's workings on ground, techniques for modeling and analyzing goals need to be applied in an actionable way. This means that ***whys*** are not just documented in a blueprint manner but rather used to direct the change by analyzing various alternative courses of action and choosing from available alternatives and reflecting the choices in affected dimensions of an enterprise.

It is our belief that is better to treat EA models as *the* version of truth and use intentional/goal models only as a technique to solve specific problems, arising particularly due to *profoundly important business strategy change drivers* (Buchanan and Soley, 2002), preserving the reasoning by translating analysis results back to EA models. This is the basis of our approach which is illustrated in Figure 1.

The EA model is presumed to be in ArchiMate while the intentional model is considered to be in i*. We choose ArchiMate for its economy of concepts and coverage of concepts pertaining to various aspects of enterprise (Sunkle et al., 2013). The next section details ArchiMate and i* metamodel mapping and how i* is applied as a problem solving technique to EA model expressed in ArchiMate. We choose i* intentional modeling among several goal-oriented requirement techniques because, first, i* has already been applied in the context of EA (Yu et al., 2006) and second, considerable literature discussing meaning and application of i* concepts is available along with several metamodels (de Castro et al., 2011) which is helpful in deciding how to map specific concepts from EA metamodel to i* metamodel. The i* metamodel used in the present mapping is adapted from (López et al., 2011).

2.1 EA and i* Metamodel Mapping

The ArchiMate generic metamodel defines active structure elements (ASEs) as the entities that are capable of performing behavior. These are assigned to behavior elements (BEs), which indicate units of activity. The passive structure elements (PSEs) are the objects on which behavior is performed (Haren and Publishing, 2012). Each of these three types and the relationships shown in Figure 2 occur in three layers of ArchiMate metamodel namely, business, application, and technology layers. A service is the externally visible behavior of the providing system, from the perspective of systems that use that service. Services are made accessible through interfaces, which constitute the external view on the active structural aspect. Thus, the business layer offers products and services to external customers, which are realized in the organization by business processes (BE) performed by business actors (ASE). The application layer supports the business layer with application services (BE) which are realized by (software) applications (ASE). The technology layer offers infrastructure services (BE) like storage and networking needed to run applications, realized by computer and communication hardware and system software (ASE) (Haren and Publishing, 2012).

The i* metamodel as illustrated in Figure 2, considers actors performing tasks as means to an end that is captured as (soft) goals (López et al., 2011). Resources may be used or created by actor while performing tasks. Actors may depend on each other to perform a task and/or to use or create a resource to achieve a goal or soft goal. Two kinds of models are used in i* namely, strategic dependency (SD) and strategic rationale (SR) models, to capture dependencies between actors and to model the intentions of actor in performing their appointed tasks respectively. In the context of an enterprise, the SD model would describe an enterprise in terms of dependencies that enterprise actors have on each other in accomplishing

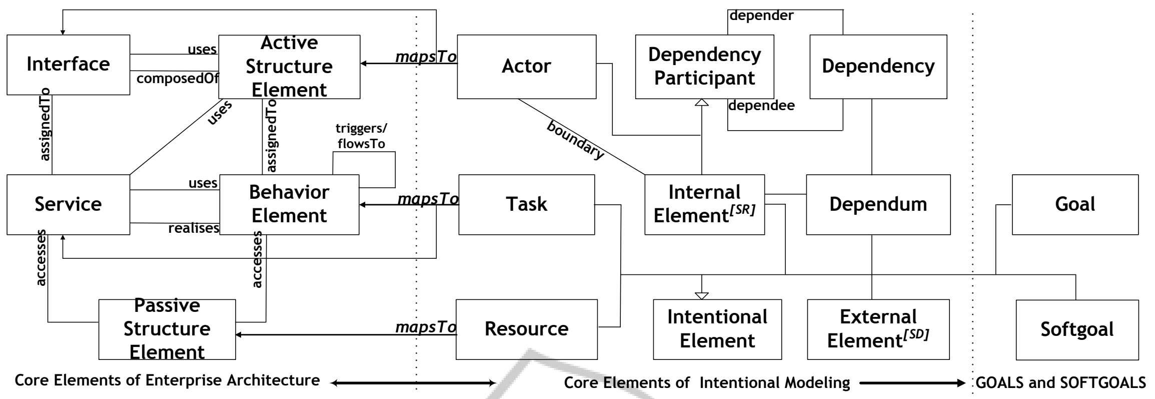

Figure 2: Mapping between generic EA metamodel and i* intentional metamodel.

their work. The SR model would describe reasoning that actors employ in determining the merit in organizing their tasks one way or the other.

As shown in Figure 2, we map the ASEs such as business actors, application components, hardware and system software, as well as interfaces to actors in i*. The BEs such as business processes, and business, application, and infrastructure services are mapped to tasks in i*. The PSEs are mapped to resources in i*, which are essentially informational or physical entities used or created by actors. Via this mapping, it is possible to say that ASEs use or create PSEs while performing BEs as means to an end that is goal(s) and/or soft goal(s).

As the general structure of models within the different layers is similar, with differing granularity and nature that is dependent on the given layer, the mappings provided between generic metamodels of ArchiMate and i* make it possible to extract an initial intentional model from the ArchiMate models of business, application, and technology layers of an enterprise. Note that cardinalities are absent in the EA metamodel due to its generic nature and thus we do not show cardinalities in the i* metamodel as well as the mapping.

In the next section, we introduce the case study that is used to elaborate EA problem solving with intentional modeling in later sections.

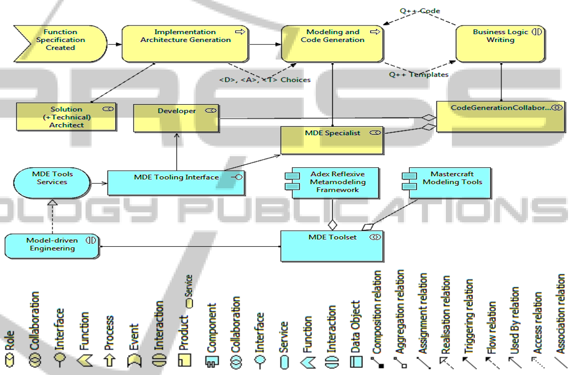

Figure 3: As-Is Enterprise Architecture Model of a Model-driven Software Development Unit.

Case Study

In this section, we re-imagine our MDE-based software development unit as an enterprise and present two distinct stages in its evolution as current and future states in retrospect.

The *first stage* in our MDE-based software development unit is characterized by the use of a unified metamodel for specifying application, database, and GUI layers of an application, model-aware Q++ language for the developers to write business logic, and separation of concerns in model-driven development for design strategies, architectural specifics, and technology platforms using aspects (Kulkarni and Reddy, 2003). The *second stage* is characterized by the use of multi-user multi-site repositories for models and code with versioning and configuration management support, component abstraction and workspaces for geographically distributed development, and change-driven development for quick turnaround of model-code operations (Kulkarni et al., 2010).

The organizational change which prompted transition from the first stage to second stage was the demand for onsite development of very large business applications. This necessitated many teams performing traditional roles of software development such as solution architecting, modeling, and developing. Furthermore, the development was carried out at geographically distributed locations with operations on same set of models and code for a given application.

We begin in the next section, by first showing a simplified EA model of the first stage of MDE-based software development unit. To restrict the scope of this case study, we capture the network of three roles in i* model namely, a solution architect (SA), an MDE specialist (MDESp), and a developer (D) in a given team. The impact of the organizational change which is large geographically distributed development is captured by asking questions of the current practices.

3 PROBLEM SOLVING WITH INTENTIONAL MODELS

The current state of the enterprise is captured in the EA model of Figure 3. Once functional specifications of given application are available, the business process of implementation architecture generation begins to which SA is assigned. SA conveys the choices of design strategies $\langle D \rangle$, architectural specifics $\langle A \rangle$, and technology platforms $\langle T \rangle$ to MDESp. $\langle D \rangle$ could be audit, persistence, caching, attribute value handling etc., $\langle A \rangle$ could be patterns of distributed architecture, middleware choices, message queuing mechanism etc., and $\langle T \rangle$ could be combinations of various technologies and frameworks that are specific to platforms as well as customer preferred technologies.

MDESp's job is to accommodate these choices in the code generators using unified metamodel. MDESp sends templates in model-aware language Q++ to D who needs to write business logic and send the templates back to MDESp for full application generation. Both MDESp and D use reflexive metamodel framework (Adex) and model processing software (Mastercraft) for carrying out their respective tasks.

Using the mapping explained in Section 2.1 and illustrated in Figure 2, we obtain an i* model of the current state of enterprise explained above. This is shown in Figure 4. Note here, that roles and application components (ASEs) from the EA model are mapped to roles and actors in i* respectively. Since Adex and Mastercraft from EA model collaborate to provide modeling service, we capture them as a single actor in i*. The business processes and application interactions (BEs) are captured as tasks and put into roles who were assigned to them in the EA model.

As more and larger application requirements came to us, we had to change the nature of teams working initially on small applications to number of subteams performing several specialized tasks. These sub-teams needed to share a single main version of models and code in order to ensure that different models are consistent with each other and business logic is consistent with the models. This prompted the need for a central repository. The first problem that we attempt to solve therefore is *how to enable secure and synchronized access to models and code for large teams?*

Problem 1 - Access by Large Teams

MDESp, D, as well as SA teams need to make use of models and code; MDESp needs access to models and metamodels, D needs to access code, and SA can search models and metamodels to make informed decision about choices of $\langle D \rangle$, $\langle A \rangle$, and $\langle T \rangle$ for current application. Evidently we need to introduce a system whose main task is to provide access to models and code for all these roles. Further requirement of this access are that concurrent multi-user operations should be enabled. In Figure 5, this is represented as a goal with two alternatives namely, using an industrial strength database for storing models and code or developing a proprietary storage system ourselves.

Of the two alternatives to achieve concurrent multi-user operations, industrial database immediately makes it possible to apply storage of models and code, whereas creating our own proprietary database tailored for storage of models and code could have taken substantial time. These contribute highly positively (make) and slightly negatively (hurt) to the soft goal of quick storage enablement. Both the alternatives would contribute slightly positively to the soft goal of secure and synchronized access differing only in how quickly the storage can be applied to our existing models and code.

Choosing either alternative leads to adding an actor in the form of a system, which we have referred to as X. In reality, this was the (multi-user) model repository system (Kulkarni et al., 2010). It can also be seen that the responsibility of performing tasks that assure secure and synchronized access to models and code is assigned to the actor X.

We observed that without explicitly stated dependencies such partitioning led to integration problems and dependencies not only between models or code but also between models and code. Additionally, we were increasingly required to develop some modules of application onsite, i.e., an application may be developed in geographically distributed manner. Here too, proper partitioning was an important issue. The second problem we attempt to solve is *how to best partition model-driven development so as to manage dependencies between models and code correctly as well as to enable geographically distributed development with this new partitioning scheme?*

Problem 2 - Distributed Development

The complete analysis of this problem is shown in Figure 6. Reader is referred to (Kulkarni et al., 2010) for a detailed overview of the development of such a

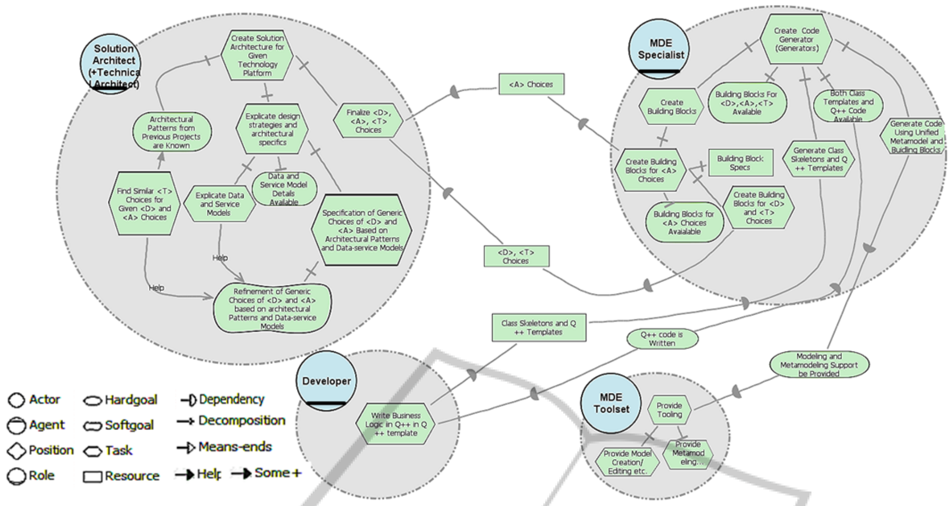
Solution Architect (+Technical Architect)
Create Solution Architecture for Given Technology Platform
Architectural Patterns from Previous Projects are Known
Explicate design strategies and architectural specifics
Finalize <D>, <A>, <T> Choices
Find Similar <T> Choices for Given <D> and <A> Choices
Explicate Data and Service Models
Data and Service Model Details Available
Specification of Generic Choices of <D> and <A> Based on Architectural Patterns and Data-service Models
Help
Help
Refinement of Generic Choices of <D> and <A> based on architectural Patterns and Data-service Models
<A> Choices
<D>, <T> Choices
MDE Specialist
Create Code Generator (Generators)
Create Building Blocks
Building Blocks For <D>,<A>,<T> Available
Both Class Templates and Q++ Code Available
Generate Code Using Unified Metamodel and Buidling Blocks
Create Building Blocks for <A> Choices
Building Block Specs
Generate Class Skeletons and Q ++ Templates
Building Blocks for <A> Choices Avaialable
Create Building Blocks for <D> and <T> Choices
Class Skeletons and Q ++ Templates
Q++ code is Written
Modeling and Metamodeling Support be Provided
Developer
Write Business Logic in Q++ in Q ++ template
MDE Toolset
Provide Tooling
Provide Model Creation/ Editing etc.
Provide Metamodeling...
Actor
Agent
Position
Role
Hardgoal
Softgoal
Task
Resource
Dependency
Decomposition
Means-ends
Help
Some+

Figure 4: i* Model of current enterprise derived from Figure 3.

partitioning scheme. Here we make remarks pertinent to the problem solving aspect of intentional modeling.

As shown in Figure 5 and now in Figure 6, the core problem without qualitative judgment is represented as a key task of the involved role/actor. This task provides a more generic description of the activity to be performed for addressing a specific problem. The specific problem or problems are represented as decomposition elements of the main task.

In the solution to the current problem, MDESp is responsible to manage model-code development to achieve partitioning of development effort and to provide support for geographically distributed development. The qualitative aspects of the desired solution are represented as soft goals, for instance in Figure 6, that dependency management be efficient and geographically distributed development be fine.

The problem of enabling geographically distributed development has hierarchy of soft goals, essentially capturing goals of goals. For instance, for the alternative of replicating model repositories for distributed development, efficient synchronization soft goal helps efficient implementation of performance intensive model operations which in turn helps fine overall support for distributed development. While on the right of Figure 6 is a (soft) goal hierarchy, immediately to the left of it, is the task hierarchy represented in the form of successive alternatives that would contribute to the (soft) goals. The result of component and workspace development (shown on the extreme left of Figure 6) is that another role (not shown here) called development manager (DM) gets the information about component and workspace from MDESp. DM uses this information to assign D to specific components and workspaces.

As solutions of multi-user multi-site repository and component-workspace abstraction were applied to problem 1 and 2, we found that teams merrily used model-code access as details of secure and synchronized access were no longer the issues. Another problem surfaced at the time. As the number of classes ranged to a few hundreds and source lines of code to a few million (Kulkarni et al., 2010), performance of various model processing operations plummeted, particularly originating in model changes. The third problem at this point that we would attempt to solve is *how to enable quick turnaround of model changes done by geographically distributed teams?*

Problem 3 - Quick Turnaround with Models

Figure 7 shows that the responsibility of the solution to this problem is given to the actor that we added in the solution of first problem, i.e., model repository system. The main task of this actor remains the same and now this actor also tries to achieve the goal of providing support for change management for large models. Alternative tasks contribute to soft goal of quick model processing operations done after a model change which in turn helps the soft goal of quick turnaround of model changes. For the details of delta modeling, reader is referred to (Kulkarni et al., 2010).

In the next section, we describe preliminary analysis on altered i* model and then transform these changes back to the original EA model.

4 FROM INTENTIONAL MODELS TO EA MODEL

Whereas the original model shown in Figure 4 has 5 dependencies, the new combined model has 11 depen-

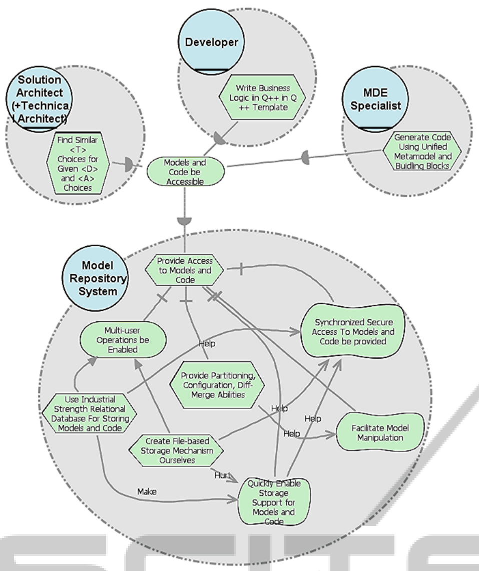
Developer
Solution Architect (+Technical Architect)
MDE Specialist
Write Business Logic iin Q++ in Q ++ Template
Find Similar <T> Choices for Given <D> and <A> Choices
Models and Code be Accessible
Generate Code Using Unified Metamodel and Buidling Blocks
Model Repository System
Provide Access to Models and Code
Multi-user Operations be Enabled
Synchronized Secure Access To Models and Code be provided
Help
Provide Partitioning, Configuration, Diff-Merge Abilities
Use Industrial Strength Relational Database For Storing Models and Code
Help
Help
Help
Facilitate Model Manipulation
Create File-based Storage Mechanism Ourselves
Hurt
Make
Quickly Enable Storage Support for Models and Code

Figure 5: Enabling secure and synchronized access to models-code for large teams.

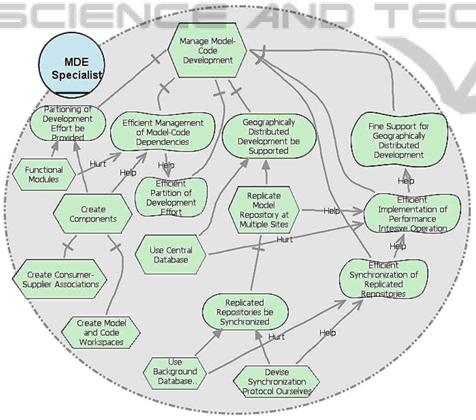
MDE Specialist
Manage Model-Code Development
Partioning of Development Effort be Provided
Efficient Management of Model-Code Dependencies
Geographically Distributed Development be Supported
Fine Support for Geographically Distributed Development
Functional Modules
Hurt
Help
Help
Efficient Partition of Development Effort
Create Components
Replicate Model Repository at Multiple Sites
Help
Help
Efficient Implementation of Performance Intesive Operation
Use Central Database
Hurt
Help
Create Consumer-Supplier Associations
Efficient Synchronization of Replicated Repositories
Replicated Repositories be Synchronized
Create Model and Code Workspaces
Hurt
Help
Use Background Database..
Devise Synchronization Protocol Ourselves

Figure 6: Partitioning for correct dependencies and distributed development.

dencies. The combined intentional model contains 22 leaves out of which 11 represent what-if alternatives specified. We used OpenOME (Horkoff et al., 2011) to carry out intentional modeling and to evaluate various alternatives.

Having chosen the better alternatives and further better sub-alternatives, with reasoning behind these choices explicit, and responsibilities of tasks to be performed clearly assigned, we can now use meta-model mapping in Figure 2 to get to EA model which captures the state that employs the solutions to the main problem of being able to support large development teams.

The EA model shown in Figure 8 builds on the model illustrated in Figure 3. Solution to **first** prob-

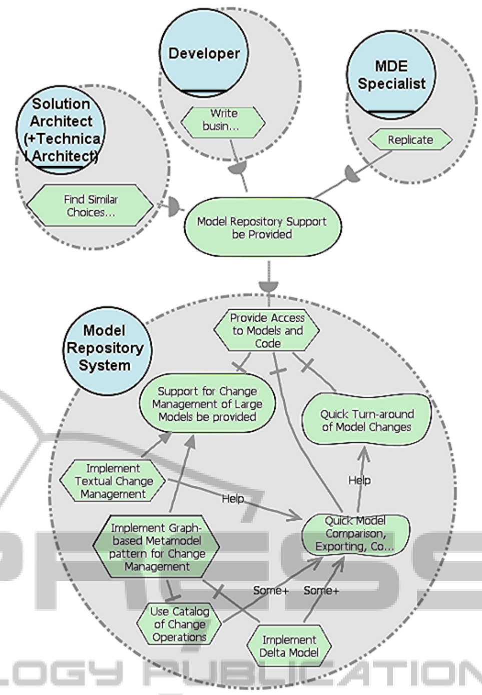
Developer
MDE Specialist
Solution Architect (+Technical Architect)
Write busin...
Replicate
Find Similar Choices...
Model Repository Support be Provided
Model Repository System
Provide Access to Models and Code
Support for Change Management of Large Models be provided
Quick Turn-around of Model Changes
Implement Textual Change Management
Help
Help
Implement Graph-based Metamodel pattern for Change Management
Quick Model Comparison, Exporting, Co...
Some+
Some+
Use Catalog of Change Operations
Implement Delta Model

Figure 7: Change management for large models.

lem adds an actor, multi-user repository system. This is represented as an application component in Figure 8 via actor-ASE mapping. This application component is assigned to the application function that carries out model-code storage and model manipulation and realizes the application service which is used by SA, D, and MDESp via repository interface.

Solution to **second** problem is implemented by MDESp in terms of partitioning of development in components and workspace, the information about which flows to DM (additional role added in the solution of second problem) who uses it in assigning D to specific components (not shown here). MDESp also carries out replication of model repositories at required multiple sites. Note that both partitioning tasks and replication tasks are represented as business process via task-BE mapping.

Solution to **third** problem involves implementing required delta model as synchronization protocol infrastructure function in the technology layer. This function realizes the infrastructure service of actual model storage and manipulation used by application service via onsite repository interface. The model synchronization protocol accesses actual infrastructure objects of models and code, which realize application objects of model repository data. These in turn realize replicated model repository instances which are accessed in business processes of all the actors.

The newly constructed EA model is actionable in the sense that it can be used for clarifying the focus

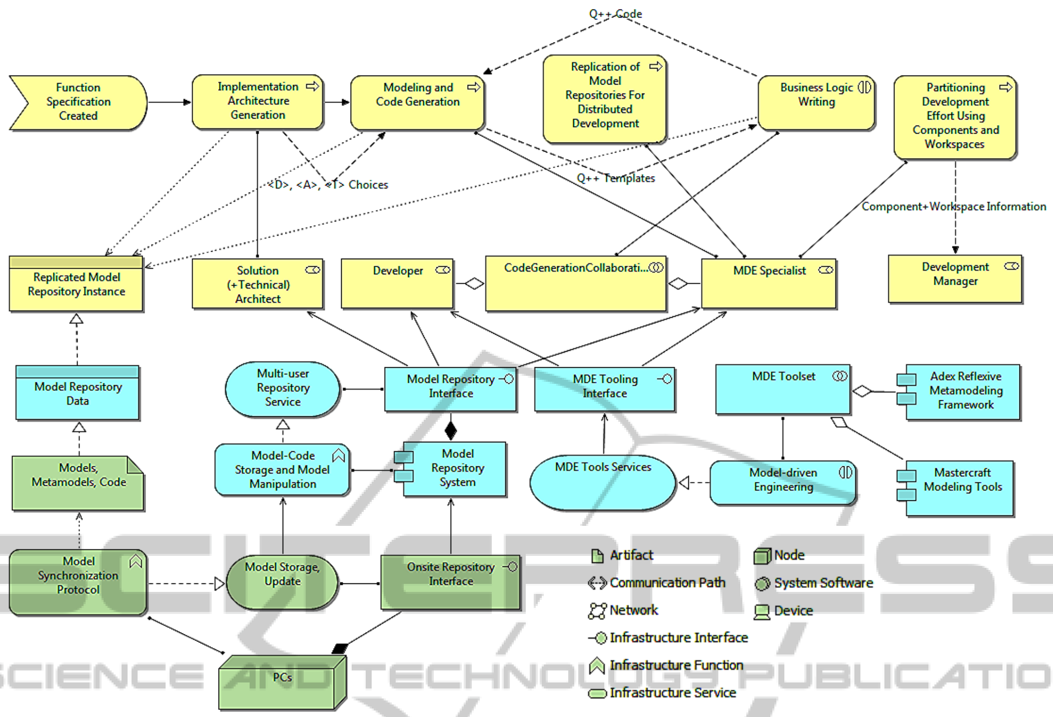
Q++ Code
Function Specification Created
Implementation Architecture Generation
Modeling and Code Generation
Replication of Model Repositories For Distributed Development
Business Logic Writing
Partitioning Development Effort Using Components and Workspaces
<D>, <A>, <T> Choices
Q++ Templates
Component+Workspace Information
Replicated Model Repository Instance
Solution (+Technical) Architect
Developer
CodeGenerationCollaborati...
MDE Specialist
Development Manager
Model Repository Data
Multi-user Repository Service
Model Repository Interface
MDE Tooling Interface
MDE Toolset
Adex Reflexive Metamodeling Framework
Models, Metamodels, Code
Model-Code Storage and Model Manipulation
Model Repository System
MDE Tools Services
Model-driven Engineering
Mastercraft Modeling Tools
Model Synchronization Protocol
Model Storage, Update
Onsite Repository Interface
PCs
Artifact
Communication Path
Network
Infrastructure Interface
Infrastructure Function
Infrastructure Service
Node
System Software
Device

Figure 8: To-be EA Model of a Model-driven software development unit (For legends other than technology layer, please refer to Figure 3).

with regards the reaction to organizational change, for instance, which ASEs are responsible to implement the solutions to problem(s) caused by the change, in what ways the solution impacts how ASEs interact and how BEs to which they are assigned get affected. Both target architectures of business and application layers are captured along with technology (hardware infrastructure) necessary to support these architectures.

5 RELATED WORK

Other researchers have proposed using intentional (Yu et al., 2006) and goal modeling (Quartel et al., 2009; Engelsman et al., 2011) concepts in conjunction with EA frameworks. Although the general approach in (Yu et al., 2006) is similar to ours, it is presumed that BMM is used on the top of existing EA (created with any EA framework) and i* is used for analysis of goals specified in BMM. Unfortunately, no mapping of any kind is given either from EA framework to BMM or BMM to i* in (Yu et al., 2006). This means that no help is provided for the modelers to make sense of EA entities in the context of intentional modeling. On the other hand, it was found that the sheer number of concepts in the language proposed in the goal modeling approach over ArchiMate (Quartel et al., 2009) made simultaneous construction of EA and goal models quite confusing (Engelsman et al., 2011) and lost the advantage of goal modeling.

We have deliberately considered generic metamodels instead of more concrete metamodels of ArchiMate[1] and i*[2]. Since the generic EA metamodel (of ArchiMate) is based on natural language concepts, it may be *possible* to use this mapping even when primary EA framework or modeling language is other than ArchiMate. This is out of the scope of this paper and first steps toward this have been taken by others as in (Berrisford and Lankhorst, 2009).

Motivation extensions in ArchiMate (Haren and Publishing, 2012) and BMM by OMG (OMG, 2010) both provide blueprint treatment of ***whys*** of an enterprise with a taxonomy of concepts related to motivation or reasons behind decisions. The taxonomical nature of these extensions is often not useful enough to use the extension as a problem solving technique but might be useful when explaining reasoning behind decisions to management.

Apart from forward and backward evaluation of

[1]Business, application, and technology layer metamodels are available in Sections 3, 4, and 5 of ArchiMate specification (Haren and Publishing, 2012) respectively.

[2]A number of i* metamodels are available (Ayala et al., 2005); we have adapted the i* core language metamodel from (López et al., 2011).

alternatives in i*, a number of other analyses are proposed such as analysis of opportunity and vulnerability, criticality and commitment (Yu and Mylopoulos, 1994), and so on, in the context of business process reengineering. In our ongoing work on analyzable enterprise models, we plan to implement such analyses via metamodel mapping proposed in this paper.

6 CONCLUSIONS

In the face of rapid changes affecting several facets of an enterprise, a holistic treatment is needed along with a mechanism to represent, analyze, and use reasons behind what enterprise is doing now and how it will face the changes. In our approach, EA provides holistic treatment of enterprise instead of point views and analysis of ***whys*** using intentional modeling reflects in all aspects of enterprise. Our initial explorations suggest that the mapping enables a clear path from as-is to to-be states of an EA model while preserving the reasoning behind every alternative chosen. Providing semantic foundation of the mapping between EA and intentional metamodel as well as automating representation and analysis of EA and intentional models is part of our ongoing work.

REFERENCES

Ayala, C. P., Cares, C., Carvallo, J. P., Grau, G., Haya, M., Salazar, G., Franch, X., Mayol, E., and Quer, C. (2005). A comparative analysis of i*-based agent-oriented modeling languages. In Chu, W. C., Juzgado, N. J., and Wong, W. E., editors, *SEKE*, pages 43–50.

Berrisford, G. and Lankhorst, M. (2009). Using archimate with an architecture method. *Via Nova Architectura*.

Buchanan, R. and Soley, R. (2002). Aligning enterprise architecture and it investments with corporate goals. *OMG Whitepaper, Object Management Group, Needham*.

de Castro, J. B., Franch, X., Mylopoulos, J., and Yu, E. S. K., editors (2011). *Proceedings of the 5th International i* Workshop 2011, Trento, Italy, August 28-29, 2011*, volume 766 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Engelsman, W., Quartel, D. A. C., Jonkers, H., and van Sinderen, M. (2011). Extending enterprise architecture modelling with business goals and requirements. *Enterprise IS*, 5(1):9–36.

Haren, V. and Publishing, V. H. (2012). *ArchiMate 2. 0 Specification*. Van Haren Publishing Series. Bernan Assoc.

Horkoff, J. and Yu, E. S. K. (2009). Evaluating goal achievement in enterprise modeling - an interactive procedure and experiences. In Persson, A. and Stirna, J., editors, *PoEM*, volume 39 of *Lecture Notes in Business Information Processing*, pages 145–160. Springer.

Horkoff, J., Yu, Y., and Yu, E. S. K. (2011). OpenOME: An open-source goal and agent-oriented model drawing and analysis tool. In (de Castro et al., 2011), pages 154–156.

Kulkarni, V. and Reddy, S. (2003). Separation of concerns in model-driven development. *IEEE Software*, 20(5):64–69.

Kulkarni, V., Reddy, S., and Rajbhoj, A. (2010). Scaling up model driven engineering - experience and lessons learnt. In Petriu, D. C., Rouquette, N., and Haugen, Ø., editors, *MoDELS (2)*, volume 6395 of *Lecture Notes in Computer Science*, pages 331–345. Springer.

López, L., Franch, X., and Marco, J. (2011). Making explicit some implicit i* language decisions. In Jeusfeld, M. A., Delcambre, L. M. L., and Ling, T. W., editors, *ER*, volume 6998 of *Lecture Notes in Computer Science*, pages 62–77. Springer.

OMG (2010). *Business Motivation Model - Version 1.1*.

Quartel, D. A. C., Engelsman, W., Jonkers, H., and van Sinderen, M. (2009). A goal-oriented requirements modelling language for enterprise architecture. In *EDOC*, pages 3–13. IEEE Computer Society.

Sunkle, S., Kulkarni, V., and Roychoudhury, S. (2013). Analyzable enterprise models using ontology. In *Proc. CAiSE Forum*. Accepted.

Wagter, R., Proper, E., and Witte, D. (2012). A practice-based framework for enterprise coherence. In Proper, E., Gaaloul, K., Harmsen, F., and Wrycza, S., editors, *PRET*, volume 120 of *Lecture Notes in Business Information Processing*, pages 77–95. Springer.

Yu, E. S. K. and Mylopoulos, J. (1994). Understanding "why" in software process modelling, analysis, and design. In Fadini, B., editor, *Proceedings of the 16th International Conference on Software Engineering*, pages 159–168, Sorrento, Italy. IEEE Computer Society Press.

Yu, E. S. K., Strohmaier, M., and Deng, X. (2006). Exploring intentional modeling and analysis for enterprise architecture. In *Tenth IEEE International Enterprise Distributed Object Computing Conference (EDOC) Workshops*, page 32.