2009 35th Euromicro Conference on Software Engineering and Advanced Applications

Foundations for a Model-Driven Integration of Business Services in a Safety-critical Application Domain

Richard Mordinyi, Thomas Moser,
Eva Kühn, Stefan Biffl
Complex Systems Design and Engineering Lab
Vienna University of Technology
Vienna, Austria
{firstname.lastname}@tuwien.ac.at

Alexander Mikula
Frequentis AG
Vienna, Austria
alexander.mikula@frequentis.com

***Abstract*—Current architectures for systems integration provide means for forming agile business processes by manually or dynamically configuring the components. However, a major challenge in the safety-critical Air Traffic Management (ATM) domain is to interconnect business services taking into account service level agreements regarding the underlying network infrastructures. In such domains, manual configuration is forbidden due to the resulting error-prone and time-consuming tasks, while dynamic configuration is not allowed due to non-deterministic decision making. In this paper we propose a model-driven system configuration approach (MDSC), which explicitly models the components of the network infrastructures and their capabilities to automatically generate a logical network configuration. Based on an industry application example, we show the feasibility of the proposed integration platform in the ATM domain and discuss the advantages and limitations.**

Keywords-systems integration, model-driven configuration, semantic requirement and capability models

I. INTRODUCTION

One of the major challenges in safety-critical environments like the Air Traffic Management (ATM) domain is the fulfillment of all functional and non-functional requirements of business services to be interconnected. In such safety-critical environments, both semantically and technologically heterogeneous services, which were originally not designed for flexible integration, need to be integrated on top of heterogeneous middleware infrastructures in a deterministic and static, but also fault-tolerant manner.

Current approaches for systems integration, like SOA, tie together services in a loosely coupled fashion in order to react on changing business requirements in an agile manner. The integration architecture for such an approach, e.g. an Enterprise Service Bus (ESB) [5], provides the means for forming composite business processes, and thus automating business functions. The configuration [18] of the components of such architecture, like the one for routing messages, can be performed either manually or dynamically [4][20]. However, the properties of a safety-critical domain [9] describe clearly how decision making has to be performed. For each decision made, a response has to be pointed out, and based on the same internal states the same decision has to result in the same output. Additionally, any missing but needed information at the time of decision making may lead to catastrophic scenarios. Therefore, an error-prone and time-consuming manual configuration of the system is forbidden, while dynamic configuration of the system is prohibited due to the possibly non-deterministic outcome of the decision process.

In previous work we introduced an overall system integration approach [12] which externalizes integration expert knowledge using semantic models and automatically derives a logical solution model with integration partner candidates, as well as a technical solution model with specific infrastructure configurations. In [13] we presented how the semantic modelling of requirements and capabilities is achieved in order to allow an automatic deduction of the logical solution model. In this paper we propose an integration platform (INTP) based on a model-driven system configuration approach (MDSC) for the ATM domain. The MDSC approach automatically derives integration system configurations from changed business requirements for INTP. Beside the business service capabilities and requirements [13], the components of the heterogeneous network infrastructures and their capabilities are explicitly modelled. Thus, the MDSC approach is capable of generating a logical network configuration independent of the underlying heterogeneous network infrastructure technologies which is used by the INTP as a clear specification how to interconnect business services.

The deployment of such a configuration is supported by an integration platform which allows a) the communication of business services over heterogeneous middleware technologies by means of an abstraction interface; b) the homogeneous binding of heterogeneous business services including the execution of derived transformation instructions to overcome the semantic gaps between business services; c) fault-tolerant, deterministic, and traceable group communication and routing. The benefits are: a) a verified systems configuration with the possibility to feed back monitoring data into the models in order to iteratively improve systems configuration; b) the separation of specification and implementation leads to a complexity and control of the integration scenario at a central point concerning only a few domain experts; c) the technical solution model can be simulated before deployment due to an automated and therefore cheap generation of integration scenarios; d) by means of automatically deriving system configurations the INTP's abstraction mechanism allows the effective reuse of existing middleware technologies; and e) configurable

978-0-7695-3784-9/09 $26.00 © 2009 IEEE
DOI 10.1109/SEAA.2009.19

coordination patterns without adding additional complexity to the INTP. Based on an industry application example, we show the feasibility of the proposed integration platform in the ATM domain and discuss advantages and limitations.

The remainder of this paper is structured as follows: section II describes a motivating industry application integration scenario; section III summarizes related work, while section IV presents the research issues. Section V describes the proposed MDSC approach, section 0 gives an overview of the developed integration platform, and section VII discusses the results of the feasibility study of the application example as well as benefits and limitations of the proposed MDSC approach. Finally, section VIII concludes the paper and identifies further research opportunities.

II. Motivating Integration Scenario Description

This section describes the integration scenario from the ATM domain used throughout this paper. The business system *Air Traffic Management Information Service* (ATMIS) has to provide information services about flights to business partners via a *Public Flight Information Portal* (PFIP). ATMIS needs to collect and refine information from at least 2 other systems: the *Central Flight Controller* (CFC) and the *Single Flight Data Processors* (SFDPs).

As shown in Fig. 1, the integration network consists of business services (e.g., CFC or ATMIS) which are connected to integration network nodes (e.g., CFC Node or ATMIS Node). Between these nodes, there may exist different kinds of network links, represented by arrows between the nodes. These links can either use different transmission technologies (e.g., radio or wired transmission) as well as different middleware technologies (e.g., SONIC[1], TIBCO[1] or IPX-based middlewares) for communication purposes. The capabilities of both kinds of technologies are explicitly modeled in order to automatically select suitable communication paths for particular service requirements, e.g., the red connection between ATMIS Node, Node Y and PFIP Node represents are reliable and secured communication path which may be requested by e.g., the ATMIS business service.

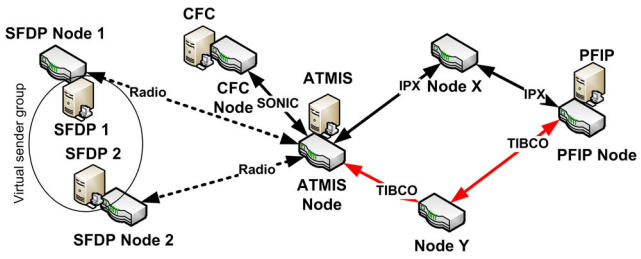

Figure 1. ATM Domain Integration Scenario

On the left hand side of Fig. 1, another feature of the ATM domain integration scenario is pictured. The business services SFDP 1 and SFDP 2 are both elements of a so-called "virtual sender group", which means that they both send the same data, but only one message should be sent to the receiving ATMIS business service. This may e.g. be the case if the SFDP business services are connected to redundant radar dishes, but only one message containing the radar data should be sent, the other is just produced for reliability purposes. So the business services need to coordinate themselves in order to determine which SFDP business service is allowed to send the message to the ATMIS business service.

[1] Progress SONIC (http://www.sonicsoftware.com) and TIBCO (http://www.tibco.com) are registered Trademarks and Commercial Off-the-Shelf (COTS) products

III. Related Work

This chapter summarizes related work on system integration technologies and gives a short overview of the model-driven architecture background of the proposed approach.

A. System Integration Technologies

System integration is the task to combine numerous different systems to appear as one big system. There are several levels at which system integration could be performed [2], but there is so far no standardized integration process that explains how to integrate systems in general.

Typical integration solutions focus only on either the heterogeneity on service level or the heterogeneity on network level. In order to cope with technological heterogeneity on service level a homogeneous middleware technology approach [7] could be used for syntactical transformation between services, while the semantical heterogeneity of services could be addressed by means of a common data schema [6]. Heterogeneity on network level may be addressed by using so called adapters transforming messages between each used combination of middleware technologies. However, in order to provide an effective continuous integration solution in this environment, both integration levels (i.e. service and network level) need to be addressed in a mutual way.

The derived limitations for such kinds of integration approaches are on the one hand the need for a common data schema [6], which is often a hard and time consuming procedure, if not even impossible in integration scenarios with several different stakeholders. On the other hand, the need for integration over heterogeneous middleware technologies with different APIs, transportation capabilities, or network architecture styles implies the development of static and therefore inflexible wrappers between each combination of middleware technologies, and thus increases the complexity of communication. Traditional approaches for integration of business services can be categorized [5] into: Hub and spoke vs. distributed integration and coupled vs. separated application and integration logic. In the following, using current technology concepts for each category a brief discussion about their advantages and disadvantages with respect to the described scenario is given.

Application servers [7] are capable of interoperating through standardized protocols, but tightly couple integration logic and application logic together. Additionally, as the name suggests a server based architecture style is used for integration and as such has proven to be inconvenient for the scenario. Traditional EAI brokers [5], some of them built upon application servers, use a hub-and-spoke architecture. This approach on the one hand has the benefit of centralized functions such as the management of business rules or routing knowledge,

but on the other hand does not scale well across business unit or departmental boundaries, although it offers clear separations between application, integration and routing logic. Message-oriented Middleware [8] is capable of connecting application in a loosely coupled manner but requires low-level application coding intertwining integration and application logic. The resulting effort and complexity of implementing an integration platform with the support for any kind of existing middleware technologies and protocols therefore is considerably high. To enable transparent service integration, the Enterprise Service Bus (ESB) provides the infrastructure services for message exchange and routing as the infrastructure for Service Oriented Architecture (SOA) [15]. It provides a distributed integration platform and clear separation of business logic and integration logic. It offers routing services to navigate the requests to the relevant service provider based on a routing path specification. Routing may be [5] itinerary-based, content-based, conditional-based defined manually [17] or dynamic [1]. In both cases the drawback is the minimal support for considering all functional and non-functional requirements of all service connections in the system. Dynamic configuration focuses mainly on creating a route for a special business case. Using manual configuration, a system integrator has to rely on his expertise, thus the high number of service interactions may get complex and the configuration error-prone. This may lead to routes that are configured in a way in which their influence on other business interactions is not fully known. As a consequence, business interactions may mutually violate their non-functional business requirements, such as message delivery within a specific time frame otherwise the message may be still useful but not up-to-date any more. Additionally, dynamic configuration may not cope with e.g. node failures fast enough due to missing routing alternatives, therefore possibly violating the same type of non-functional business service requirements.

B. Model Driven Architecture Background

The major goal of the Model Driven Architecture (MDA) approach is the separation of system functionality specification and implementation [11]. Using the so called Computation Independent Model (CIM) the MDA framework can be used to construct the models [11] describing system requirements and behaviour in a formal way, like e.g. by using UML. The separation of system functionality and implementation specifications is modelled in the Platform Independent Model (PIM), which is refined out the CIM, normally by hand [11]. The main functionality of the PIM is to specify system structure and behaviour independently of the platform it is deployed to. The separation of system functionality on a specific technology platform is described in the Platform Specific Model (PSM), which is the result of the PIM transformation to the target platform. The advantages [10] of the MDA framework are (1) automated generation of results improving productivity, development duration, and cost; (2) focusing on the creation of conceptual models rather than on logical and technical details; (3) developed transformations can be reused; (4) adaptations due to changes of the target platform need to be made in the PIM only; and (5) new requirements defined in the CIM are passed to PIM and PSM immediately and therefore changes are reflected automatically [11]. The MDSC approach described in this paper is similar to the approach presented in [3]. Based on requirement and capability models which represent documents, integration expert knowledge and estimation/measurements of the integration network capabilities [12], a logical solution model which represents the set of suitable integration partners is derived automatically [13]. This logical solution model then is transformed into the technical solution model, representing the specific integration configuration for the underlying integration network technologies.

IV. RESEARCH ISSUES

Recent projects with industry partners from the safety-critical ATM domain raised concerns about the automated configuration of modern technology-driven integration environments. For efficient and effective system integration a major goal was to improve the capability of engineers to automatically derive an integration solution configuration by facilitating team work and tool support. Consequently, we propose a model-driven approach that uses explicitly modelled requirements and capabilities of business services and network infrastructures for deduction and deployment of an integration solution configuration. Therefore, this paper focuses on the following research issues:

RI-1: Model-Driven System Configuration. To what extent is the proposed MDSC approach feasible and requires less effort for integration scenarios compared to a traditional manual process? How does measured runtime data that is fed back into the used capability models, improve the accuracy of the configuration and thus of the integration platform? How do investment efforts and improvements evolve with the number of repeated configurations?

RI-2: Integration Platform. To what extent can the introduced integration platform reduce the complexity of integrating business services? How can the introduced abstraction interface reduce coordination complexity by means of configuration only?

In order to show the feasibility of the proposed MDSC approach we gathered requirements and capabilities from an industry application example and conducted the process steps. The results of the evaluation are discussed in chapter VII.

V. MODEL-DRIVEN SYSTEM CONFIGURATION

This section describes the model-driven system configuration (MDSC) process [12]. As shown in Fig. 2, the MSDC process consists of 5 major process steps. In the following paragraphs, these steps are explained in detail, with regard to the example data presented in the figure.

Business Services in the ATM domain. For each legacy information system to be integrated, the subject matter expert (SME) responsible for the particular system describes the messages which are either provided or consumed by the business services provided. Both the structure (i.e., data types and semantic meanings) and the format of the exchanged messages are described [13].

In the example, there are 2 business services shown on the on the left hand side, "CFC" and "ATMIS". Additionally, 4 message types are presented using a tuple-

based notation. The "CFC" service provides the "CFC-Message" consisting of information about a certain flight, i.e., flight number, departure and destination airport, planned time of departure and estimated duration of the flight. In contrast, the "ATMIS" service consumes the "ATMIS Message" consisting of additional flight information compared to the "CFC-Message", namely the estimated day and time of arrival of a certain flight.

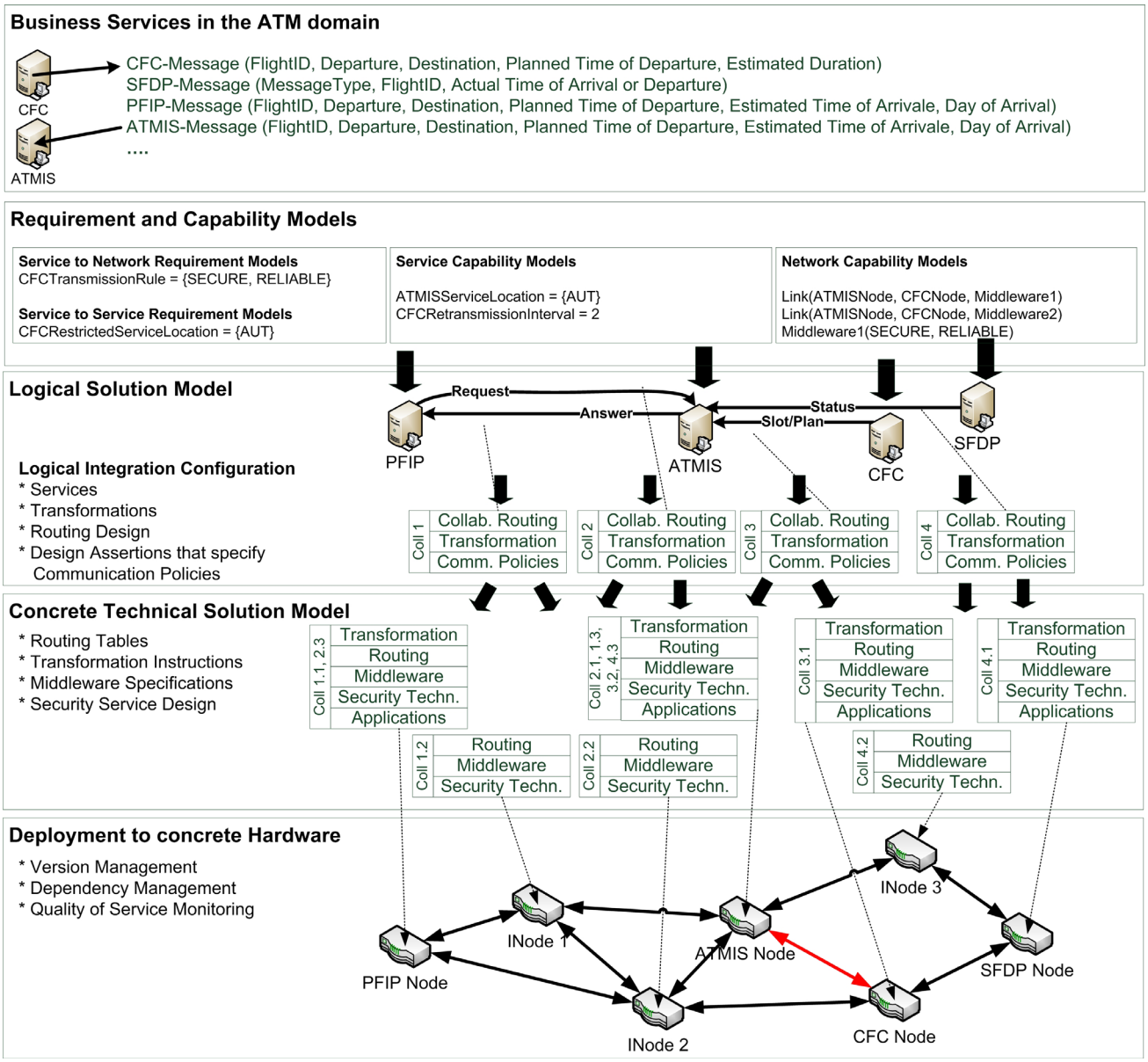

Figure 2. MSDC process overview

Requirement and Capability Models. In addition to the description of the provided or consumed message of the participating business services, the business services define extra requirements regarding either possible integration partner candidates (i.e., other business services) or the underlying heterogeneous network infrastructure [12].

The example shows three different kinds of requirement and capability models. On the left hand side, the requirements of the "CFC" service regarding both the transmission using the underlying heterogeneous network infrastructure (e.g., the transmission needs to be secure and reliable) as well as the required capabilities of possible integration partners (e.g., an integration partner services has to be an Austrian service) are presented exemplary. In the middle, the capability models of business services are shown exemplary. E.g., the "ATMIS" service has a defined service location of "Austria", and the "CFC" service has a defined retransmission interval of 2 seconds. On the right hand side, the capability models of the underlying integration network infrastructure are presented. The integration network consists of nodes and links connecting these nodes. Each link presents a specific middleware technology and may define additional capabilities of this middleware, e.g., the capability of "Middleware1" to support secure and reliable transmissions. In comparison to a traditional integration process, the outcome of the so far described process steps is a set of machine-understandable knowledge models describing both the message structures as well as the requirements and capabilities of the legacy

information system to be integrated and the capabilities of the underlying integration network infrastructure.

Logical Solution Model. The externalized knowledge which is captured in the knowledge models created in the previous steps is used to automatically derive the set of possible integration partners using ontology-based reasoning, allowing an easier and less error prone identification of possible integration partners compared to the traditional integration process [12][14].

Based on the legacy system descriptions, the description and mapping of the domain knowledge and the description of the architecture and capabilities of the integration network, the possible sending and receiving service partners are derived using heuristics and ontology-based reasoning .In the example, this is represented as a graph consisting of the possible collaborations (i.e., the services which are able to communicate). As shown in Fig. 2, there are 4 automatically derived collaborations: collaboration 1 between "ATMIS" and "PFIP", collaboration 2 between "PFIP" and "ATMIS", collaboration 3 between "CFC" and "ATMIS", and collaboration 4 between "SFDP" and "ATMIS".

Concrete Technical Solution Model. Based on the logical solution model derived in the previous process step, the technical solution model for each integration node is generated automatically. This technical solution model is an XML configuration which is interpreted by the integration platform introduced in chapter 6. The major components of the technical solution model are a) routing tables that specify where certain received messages belonging to a specific collaboration should be forwarded to - there are more than one routing targets for a specific collaboration (so called "backup routes"), which are automatically used in case of unavailability of the original target integration node; b) transformation instructions that define how messages originating from business services should be transformed before sending them to other business services via the integration nodes; c) middleware specifications that define the configuration parameters and access methods for each connected specific middleware technology of a particular integration node; d) application specifications that define the configuration parameters and access methods for each connected business service of a specific integration node; and finally e) security specifications of the particular integration node (i.e., encryption protocols or certificates to use for the transmissions).

As shown in Fig. 2, the concrete technical solution model for each single integration node contains information about all collaborations which use this particular node. Additionally, there is a difference between integration nodes that are connected to business services and integration nodes without connected business services (so called "intermediate nodes"). While the technical solution model of intermediate nodes only contains routing tables, middleware specifications and security information, the technical solution model of integration nodes connected to business services additionally contains transformation instructions and application specifications.

Deployment to concrete Hardware. Finally, the concrete technical solution model for each single integration node is deployed to the particular integration platform (see chapter VI).

VI. INTEGRATION PLATFORM

The MDSC process results in a solution model that needs to be deployed. Additionally, the process is capable of improving the system's configuration by means of monitoring data collected during execution. In the following the integration platform for the MDSC approach is described.

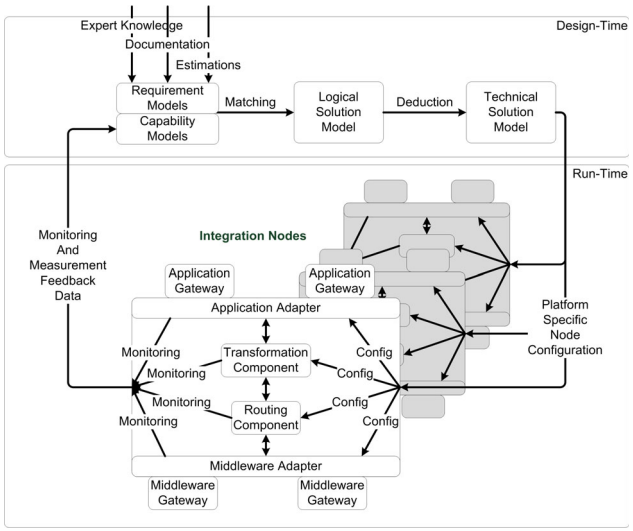

Figure 3. Integration Platform

The main task of the integration platform (INTP) (see Fig. 3) is to interconnect business services a) by binding business services and middleware technologies; b) by routing messages in a fault-tolerant manner with respect to virtual sender groups over heterogeneous middleware technologies; and c) by transforming messages to overcome the semantic gaps between business services. The INTP is installed on every node described in the network capability model and uses the derived solution model to configure its components:

Application Adapter. Based on the automatically derived configuration, the adapter loads the so called Application Gateways (AGs). An AG represents the connection to the business service and is implemented by the developers of the service. Similar to JBI [19], by means of the AG the service is capable of sending and receiving messages asynchronously. The interfaces of the Application Adapter and of the AG do not need to be described by means of WSDL since their capabilities have already been defined in the capability models. Messages sent by the AG additionally receive a so called CollaborationID representing the collaboration that has been calculated in the Logical Solution Model. This ID helps the Routing Component to route messages (i.e., the Routing Component looks up the specific network route for a particular CollaborationID).

Middleware Adapter. Based on the automatically derived configuration, the adapter loads the so called Middleware Gateways (MGs). An MG represents a communication link with a specific middleware technology between two nodes only. If there are several different communication links between two nodes, then there has to be one MG available for each possibility. The MG functions as a wrapper and knows how to interact with the real middleware that is actually forwarding the

message to the destination specified by the routing component. The responsibility of the MG is to operate and to optimize the middleware technology according to the capabilities which have been specified in the capability models and selected during the derivation of the logical solution model. Similar to the AG, the interface of the MG consists of methods for sending and receiving messages asynchronously.

Transformation Component. The solution model provides transformation instructions specifying how to handle certain message types. The instructions have been derived from the requirement and capability models describing the services and thus the Transformation Component is able to manipulate message structure and content accordingly. The component can change message data types, split messages into several segments, merge different segments into a message, replace or enrich certain information of a message, or perform any combination of the described possibilities.

Routing Component. The automatically derived configuration of the routing component contains a routing logic specifying where to forward a message. A message can be either forward to a local service via the Transformation Component or to one of the installed MGs. Consequently, the routing logic specifies either the target AG or the target MG which is used to send the message to the next hop along the route to the final target service. The chosen MG has a priori been selected during the derivation of the concrete technical solution model. Additionally, the solution model contains several other routes, so called "backup routes", which have been calculated during the derivation of the concrete technical solution model. These backup routes are used by the routing logic of the integration platform if the originally targeted next hop is not available any more. This allows the integration platform to react on changing network conditions quickly.

The scenario in chapter II introduced the problem of virtual sender groups and the need for coordination between the participating members of that group. To keep the abstraction interface reduced to the methods send and receive and the implementation of the integration platform less complex, it has been avoided to add a component responsible for group communication only. Additionally, traditional group communication mechanisms are not capable of coordination over multiple heterogeneous middleware technologies. Therefore, the interface is configurable as well, and as such the send method can be intercepted by aspects representing the appropriate strategy for coordination [16]. The need for a virtual sender groups is derived from the data in the service capability and requirement models. Additional collaborations with unique collaboration-IDs are set up between the members of a virtual sender group, and in the concrete technical solution model an appropriate route between those nodes is calculated. This means that the aspect is configured with information about the virtual sender group and which collaboration-ID it has to use to reach other virtual sender group members. When the aspect receives a message from a business service that is a member of the virtual sender group, it withholds the message until the group has reached a decision. Either the message is then sent via the original MG or discarded.

The task of the **Monitoring Component** of the integration platform is to collect information (like transmission speed and maximum bandwidth between two nodes, the number and size of exchanged messages, the time needed to reach an agreement in a virtual sender group, or the number of node failures resulting in accurate failure probability values) that may help to improve the capability models reflecting a more realistic description of the network infrastructure and the business services. This results in a configuration that is adapted to the real environment.

VII. DISCUSSION

Within a research project with two industry partners, the MDSC approach and the integration platform have been evaluated by means of several different scenarios from the ATM domain. In this chapter we discuss the benefits and limitations of the proposed approach with special respect to the introduced research questions (see chapter IV).

Model-Driven System Configuration. In [12][13] the integration process, MDSC is based on and which uses requirement and capability models, has been evaluated in comparison to traditional integration processes. Major results were that the proposed approach took considerably shorter for the modelling phase and lowered the risk of errors in the system configuration due to automated model checking using ontology-based reasoning. Additionally, traditional model checking approaches are focusing primarily on single models (like UML), but they lack support for checking heterogeneous or integrated models resulting in error-prone and time-consuming human effort. The approach has proven to be especially suitable for integration scenarios with frequent reconfiguration due to changing business requirements or network infrastructure. This allows manipulating capability and requirement models in order to simulate integration scenarios for fine-tuning of business interactions. The benefit arises from the option to cheaply generate system versions that can be analyzed to better understand the trade-offs of different capabilities in the case study context, e.g., the valuation of different middleware technologies on the distribution of traffic in the system. Additional advantage of the approach is that the complexity of manipulating models and as consequence the solution model for the integration platform is focused at a central point that can be managed by a few experts only. In the traditional integration process, administrators have just a partial view of the entire system and may try to optimize their business interactions locally. This may result in an overall system behaviour that maybe was not intended. However, the proposed MDSC approach always tries to optimize the integration scenario with a global view over the entire system. Compared to traditional high-level MDA based approaches, the MDSC approach is adapted to a specific domain (like ATM), resulting in a skipped CIM, and directly in a PIM that is not very high-level.

The collection of monitoring data from integration platform allows a) comparing the described capability models with the real behaviour of the system and b) updating the existing values of the capability models automatically based on the measured real life data. Additional monitoring data, such as availability of nodes,

which can only be estimated in the first place helps to precise the capabilities of the network infrastructure. By including this information into the calculation of routes, the overall dependability of the integration solution is improved, since probabilistic factors are used in the models.

Integration Platform. The automatically derived configuration predetermines the overall behaviour of the integration scenario. The more specifications the configuration contains the better the integration platform can react on changing circumstances. This allows the integration platform to work in a predefined deterministic way without "surprises" during execution regarding e.g. network failures, service failures, or network bottlenecks. The complexity of integrating business services is shifted away from the integration platform (run-time) to the MDSC approach at design time, minimizing the time criticality of the integration solution. This allows keeping the implementation of the integration platform itself as simple as possible since it is entirely dependent on configuration instructions only. As proof-of-concept we have implemented a prototype of the integration platform making use of a plug-in architecture style to enforce runtime policies based on the requirement and capability models.

The interface abstracting the heterogeneous middleware technologies allows injecting aspects if defined by the configuration used to coordinate services if they are in a virtual sender group. Additionally, aspects are configured by means of different strategies representing different ways to reach a decision in a group. Adding the possibility to intercept communication methods in the integration platform and to configure them by the MDSC additionally minimizes the complexity of the integration platform implementation.

Compared to traditional integration solution the middleware adapter abstracts any kind of middleware technologies. While in traditional solutions connectors between each used combination of different middleware technologies need to be implemented, the integration platform requires only the binding to the interface of the middleware adapter only. Although the approach of a common interface is not sophisticated, the benefit of it is a common interface with different transmission semantics. The semantic of the method, e.g. reliable or secure communication, depends on the capability of the middleware that is represented by that interface.

VIII. Conclusion and Future Work

In this paper we proposed and discussed a model-driven system configuration approach (MDSC) for the ATM domain, which explicitly models the components of the heterogeneous network infrastructures and their capabilities to automatically generate a logical network configuration. Based on this logical solution model, a technical solution model is derived automatically and deployed using an integration platform. This enables the communication of business services by means of an abstraction interface which bridges the underlying heterogeneous middleware technologies, additionally semantic gaps between business services are bridged by executing automatically derived transformation instructions. Furthermore, the model-driven approach allows fault-tolerant, deterministic and traceable group communication and routing mechanisms.

Based on an industry application example, we discussed the benefits of the MDSC approach and the integration platform. Major results of the discussion are: a) the manipulation of capability and requirement models allows the efficient generation of integration scenarios for fine-tuning business interactions, in order to better understand the trade-offs of different integration solutions; b) the complexity of the solution model is focused at a central point and therefore manageable by a few experts only; c) monitoring data from the integration platform can be compared with the described capability models respectively fed back into the capability models, updating the existing values with measured real-life data; d) the abstraction mechanism allows the effective reuse of existing middleware technologies by means of automatically deriving system configurations, furthermore, the access to interoperable heterogeneous middleware technologies is automatically derived without the need for human intervention for configuration; e) only basic methods (e.g., sending and receiving messages) of the underlying middleware technology are used in the abstraction interface, additional functionality (e.g., reliable transmission) is modelled in the capability model, used for derivation of the logical solution model and configured automatically by the technical solution model; and f) the abstraction interfaces allows the injection of aspects defined in the configuration, minimizing the complexity of the integration platform implementation.

Further work will include a large-scale evaluation of the proposed MDSC approach using scenarios and traditional integration effort measurements of a real-world integration project. Further work will also include a mechanism to describe a set or combination of different characteristics in the capability model and use this information during the derivation of the logical solution model, and configure the middleware technology accordingly using the derived technical solution model. Another current open research issue is the deployment of and the synchronized switch to a new version of the solution model. Finally, tool supported mechanisms to automatically feed back measured runtime information into the capability models need to be developed.

Acknowledgment

The authors would like to thank all members of the SWIS project performed 2006-2008 at Vienna University of Technology together with Frequentis AG and Austro Control GmbH.

References

[1] X. Bai, J. Xie, B. Chen, and S. Xiao. Dresr: Dynamic routing in enterprise service bus. In ICEBE '07: Proc. of the IEEE Int. Conf. on e-Business Engineering, pages 528–531, Washington, DC, USA, 2007. IEEE Computer Society.

[2] K. Balasubramanian, A. Gokhale, G. Karsai, J. Sztipanovits, and S. Neema. Developing applications using model-driven design environments. Computer, 39(2):33, 2006.

[3] S. Biffl, R. Mordinyi, and A. Schatten. A model-driven architecture approach using explicit stakeholder quality requirement models for building dependable information systems. In Proc. of 5th Intl. Wsh. on Software Quality, pages 6–6, May 2007.

[4] C. Cappiello, M. Comuzzi, and P. Plebani. On automated generation of web service level agreements. Advanced Information Systems Engineering, pages 264–278, 2007.

[5] D. Chappell. Enterprise Service Bus. O'Reilly Media, Inc., 2004.

[6] A. Halevy. Why your data won't mix. Queue, 3(8):50–58, 2005.

[7] G. E. Harris, D. Leibs, J. Carri`ere, F. Nagy, J. Crupi, and M. Nally. Application servers: one size fits all...not? In OOPSLA'03: Companion of the 18th annual ACM SIGPLAN Conf. on Object-oriented programming, systems, languages, and applications, pages 284–285. ACM, 2003.

[8] G. Hohpe and B. Woolf. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Longman Publishing Co., Inc., 2003.

[9] H. F. Lipson, N. R. Mead, and A. P. Moore. Can we ever build survivable systems from cots components? In Proc. of the 14th Int. Conf. on Adv. Information Systems Engineering, pages 216–229, London, UK, 2002. Springer-Verlag.

[10] J.-N.Mazon, J. Trujillo, M. Serrano, and M. Piattini. Applying mda to the development of data warehouses. In DOLAP '05: Proc. of the 8th ACM Int. Wsh. on Data warehousing and OLAP, pages 57–66. ACM, 2005.

[11] S. J. Mellor, S. Kendall, A. Uhl, and D. Weise. MDA Distilled. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.

[12] T. Moser, R. Mordinyi, A. Mikula, and S. Biffl. Efficient system integration using semantic requirements and capability models. In Proc. 11th Int. Conf. on Enterprise Information Systems, 2009.

[13] T. Moser, R. Mordinyi, A. Mikula, and S. Biffl. Making expert knowledge explicit to facilitate tool support for integrating complex information systems in the atm domain. In Proc. of the Intl. Conf. on Complex, Intelligent and Software Intensive Systems, 2009.

[14] T. Moser, K. Schimper, R. Mordinyi, and A. Anjomshoaa. Samoa - a semi-automated ontology alignment method for systems integration in safety-critical environments. In Proc. of the 2nd IEEE Intl. Wsh. on Ontology Alignment and Visualization, 2009.

[15] M. P. Papazoglou and W.-J. Heuvel. Service oriented architectures: approaches, technologies and research issues. The VLDB Journal, 16(3):389–415, 2007.

[16] D. Powell. Group communication. Commun. ACM, 39(4):50–53, 1996.

[17] F. Satoh, Y. Nakamura, N. K. Mukhi, M. Tatsubori, and K. Ono. Methodology and tools for end-to-end soa security configurations. In SERVICES '08: Proc. of the 2008 IEEE Congress on Services - Part I, pages 307–314, 2008.

[18] M.-T. Schmidt, B. Hutchison, P. Lambros, and R. Phippen. The enterprise service bus: making service-oriented architecture real. IBM Syst. J., 44(4):781–797, 2005.

[19] R. Ten-Hover and P.Walker. Java Business Integration (JBI) 1.0. Sun Microsystems, Inc., 2005.

[20] G. Ziyaeva, E. Choi, and D. Min. Content-based intelligent routing and message processing in enterprise service bus. In ICHIT '08: Proc. of the 2008 Int. Conf. on Convergence and Hybrid Information Technology, pages 245–249, 2008.