Three Keys to Developing and Integrating Telecommunications Service Management Systems

Vincent P. Wade, Trinity College Dublin
David Lewis, University College London

ABSTRACT With the deregulation of the telecommunication industry in Europe and the United States, communication and information services (e.g., multimedia entertainment services, educational services) are being increasingly delivered across value chains of network, service, and value-added service providers. The benefit of such interoperable services is the provision of "one-stop shopping" in which "tailored" services are delivered without their customers dealing with the multiplicity of underlying telecommunication services and network providers. The difficulty with such delivery chains is the complexity of managing these services across the different provider organizations (i.e., across both administrative and technological domains). These difficulties include achieving an understanding of business process across the organizations and the heterogeneity of the components to be (re)used to support these business processes in the organizations. This article examines three crucial elements in ensuring successful and flexible development of such service management systems — namely, a development process which is customized to support management system component development and component reuse; the development of business models capable of representing the underlying business processes for these systems; and an integration strategy designed to assist the flexible and timely cooperation of these management components both within a single organization (single administrative domain) as well as across organizations (multidomain). trends, proposes an integrated approach to realizing successful multidomain management systems. The proposed three strands consist of a customized management development process, a (workflow-engine-based) integration technology, and a business process modeling technique. Finally, the article draws conclusions based on experience implementing the proposed approach.

In the liberalized telecommunications market, which is occurring in both Europe and the United States, the integration of management functionality is a critical property of operational support systems. Such integration of management systems must occur over multiple organizational and technological boundaries (known as *multidomain management systems*). The scale, heterogeneity, and geographic distribution of network and service management systems across the value chain of network and service providers present great difficulties in the fulfillment, ensuring, and billing of new services. For example, the Network Management Forum (NMF, recently renamed as TeleManagement Forum, TMF) conducted a survey involving the principal public network operators in Europe. It identified the main business drivers for these service providers as cost reduction in provision of services, improved process flow across provider organization management systems, greater management process automation, greater customer management control, improved quality of service (QoS) management, and greater range of management services.

This article concentrates on three key aspects of implementing management systems, which address the above requirements, namely:

- A development process to support the design and implementation of reusable management components
- An integration technology to allow management components to be "sequenced" to achieve specific business objectives
- A representation of business process(es) which describe how specific business objectives are realized

The article provides an investigation of the emergent trends for these elements and, based on an analysis of these

Element #1: A Development Process for Management Components and Systems

The ability to easily integrate new or changing business processes is a very important advantage, which can be exploited by network operators, service providers, and equipment vendors. In this complex multidomain situation, the key to integration is to ensure the effective flow of information and controlled interaction between management systems. This requires that the development process adopted by the management system developers support both information flow and interaction across systems rather than being considered standalone applications, as is often the case with existing management applications. It must be recognized that realization of such large-scale (integrated) management systems relies on the effective reuse of well understood patterns of design, component designs, and implementations. Effective component reuse techniques are required to realize the widely held industry belief that bespoke management systems are too costly to develop or maintain and too slow to adapt to changing market needs.

Trends in the Development of Management Components and Systems

This section identifies recent significant developments in management software engineering. The late '80s and early '90s saw a rise in usage of many different object-oriented (OO) analysis and design techniques. Principal among the "second generation" of these methodologies are Rumbaugh's Object Modeling Technique (OMT), Jacobson's OO Soft-

0163-6804/99/$10.00 © 1999 IEEE

ware Engineering (OOSE), and Booch's Object Oriented Analysis and Design (OOAD) methodology. Each of these methods had their own unique (diagrammatic) notation, design process, and computer-aided software engineering (CASE) tools that support both these notations and design processes (sometimes). The current trend in OO design notation is to harmonize existing approaches rather than develop brand new modeling techniques. This has been realized as the standardization of a Unified Modeling Language (v. 1.0 released March 1997). UML is a set of modeling notations, which are independent of any software development process. It specifies the modeling notation and the semantics underlying this notation. The notations are claimed to be backward compatible to these three principal modeling approaches [1]. The Object Management Group (OMG) has recently accepted UML as a set of standard modeling notations.

However, the distinction between a development process (method or methodology) and a modeling language is important. The development process is an explicit way of structuring one's thinking and actions. It tells the user (developer) how to do it, when to do it, and why it is done. A modeling language is a set of rules directing how to use (one or more) notations. UML itself is independent of any particular development process. It is envisaged that development processes will be tailored for specific application domains [2].

Trends in Telecommunications Management Systems Development — Several bodies have already addressed problems in the areas of service management. The telecommunications management network (TMN) architecture (M3010) defines reference points to indicate interfaces between implementations of the functional units. The TMN family of standards also includes methodological guidance on the development of management interfaces (M3020). Initially it was assumed that these interfaces would be implemented using open systems interconnection (OSI) management — that is, Common Management Information Protocol (CMIP) used to access managed objects defined in Guidelines for the Definition of Managed Objects (GDMO); however, current work by International Telecommuinication Union (ITU) Study Group 4 is looking into OMT- and UML-based representations of management information and systems.

The TMF is an industrial forum which aims to build on existing TMN standards specifying procurement guidelines that directly reflect the industry's short-term needs. It has developed a business process model that, based on surveys of major service providers, provides a more detailed breakdown of the typical business processes involved in a service provider's operations management activities [3]. The TMF has also produced its own internal guidelines for developing agreements on management interfaces [4]. This is similar to a general system development approach, and draws on analysis techniques such as use cases and graphical modeling techniques such as OMT class and sequence diagrams.

Another body that has performed in-depth studies of service management is the Telecommunication Information Network Architecture Consortium (TINA-C). Notable results by this group in relation to open service management systems have been the development of a general business model and reference point scheme, the detailed modeling of specific service management functions, and the application of reusable OO components in the architecture. TINA-C has developed internal guidelines for modeling its systems. These are based on the five open distributed processing (ODP) viewpoints that separate enterprise, computational, informational, engineering, and technology concerns. This has been supplemented with OMT class diagrams, sequence diagrams, and simple block diagrams showing computational component structures and their interfaces.

Under the European Union Advanced Communications Technologies and Services(ACTS) research program, several projects have studied the issue of development processes for management systems. The PROSPECT project implemented a series of multidomain management systems in phases over three years, with the aim to reuse and evolve components between phases. A use-case-driven development methodology was followed which, besides use cases, also employed class, sequence, collaboration, and component diagrams. The process was applied to the analysis of multidomain management processes and to the complete development cycle, from analysis to testing, of both single providers' systems as well as individual reusable components. The process also formed the basis of the ACTS guidelines on the design of multidomain management systems [5].

Element #2: Integration of Components

A second element in achieving a successful multidomain management system composed of existing systems and reusable off-the-shelf components is the growing area of integration techniques and technologies. Integration technology is required which allows the flexible and timely coordination of component interactions to achieve specific business (i.e., management) goals.

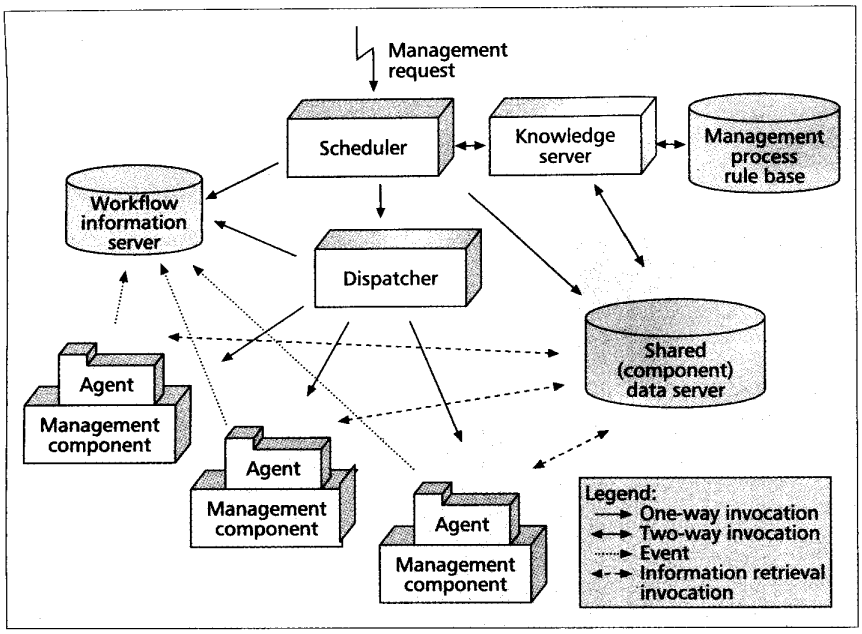
Options for Component Integration

Recent years have also seen the almost standard use of gateway systems to provide integration of heterogeneous systems, such as the joint TMF/XOpen gateway specification for CMIP, Simple Network Management Protocol (SNMP), and Common Object Request Broker Architecture (CORBA). This gateway technology has proven attractive to both telecommunications operators and network element vendors and has led commercial productization (e.g., UH Communications' CORBA-to-CMIP gateway). Different approaches to the use of gateway integration between heterogeneous management architectures have been further explored in the ACTS guidelines for TINA and TMN coexistence [6]. Other component integration technologies are based on the notions of introspection, reflection, and event models (e.g., Microsoft's Active X Controls, Javasoft's JavaBeans, and the OMG's CORBA Component Model — currently in the voting phase).

Another approach is the use of workflow technology to provide "business process" execution. The advantage of the workflow approach is that it not only provides component integration, but also supports the entire "business process" since it is executed as a sequence of component executions and interactions. The workflow *enactment engine* separates the business logic of which and when activities are sequenced and enacted in order to satisfy some business objective. This form of component integration provides the flexibility in which entire business processes can be revised, extended, or enhanced without disturbing the components. Another advantage of the workflow engine approach is that the business process in execution can be automatically monitored and managed.

A Telecommunications Management Workflow Engine

This article proposes the use of a workflow engine for telecommunication management. There is a growing aware-



■ **Figure 1.** *An architecture for a telecommunications management workflow engine.*

ness that workflow management tools and techniques may provide a vital element in the coordination of distributed components within different provider domains while allowing greater flexibility and the necessary degree of autonomy [7]. Indeed, workflow-based approaches have already proven successful in other application areas (e.g., manufacturing and office systems). More specifically, workflow systems have shown that they can improve efficiency, leading to lower costs or higher workload capacity; improve control, resulting from standardization of processes; and improve the ability to supervise processes.

In this article we describe a telecommunications management workflow engine (TMWE) which coordinates and executes management processes through the execution of a *management process rule base*. The TMWE consists of a scheduler, a dispatcher, a knowledge server, and several information servers.

The scheduler accepts management requests and initiates instances of these management processes. The scheduler uses a knowledge server to interrogate the management process rule base and determines the next activity to be enacted. The scheduler is implemented as a multithreaded process in order to deal with concurrent management requests. When the scheduler initiates work, this work is logged within a workflow information server (WIS). This WIS server maintains the state of all management process instances (i.e., all instances of management requests currently being executed within the management system). Once the next activity to be enacted to implement a management process has been identified, the scheduler passes this information to the workflow dispatcher.

The dispatcher is responsible for the invocation of the appropriate management component to support this activity. Figure 1 depicts the TMWE and illustrates the components in the system.

The dispatcher is likewise multithreaded to support concurrent component invocations. Typically, before a component can be invoked, some input parameters have to be retrieved. Such parameters may be configuration information or outputs from the execution of other component invocations. The dispatcher could potentially become congested if it must perform this information gathering as well as carry out concurrent component invocations. Also, the implementation of the dispatcher may become very complex if it has to know or interpret the information (parametric) requirements of each component.

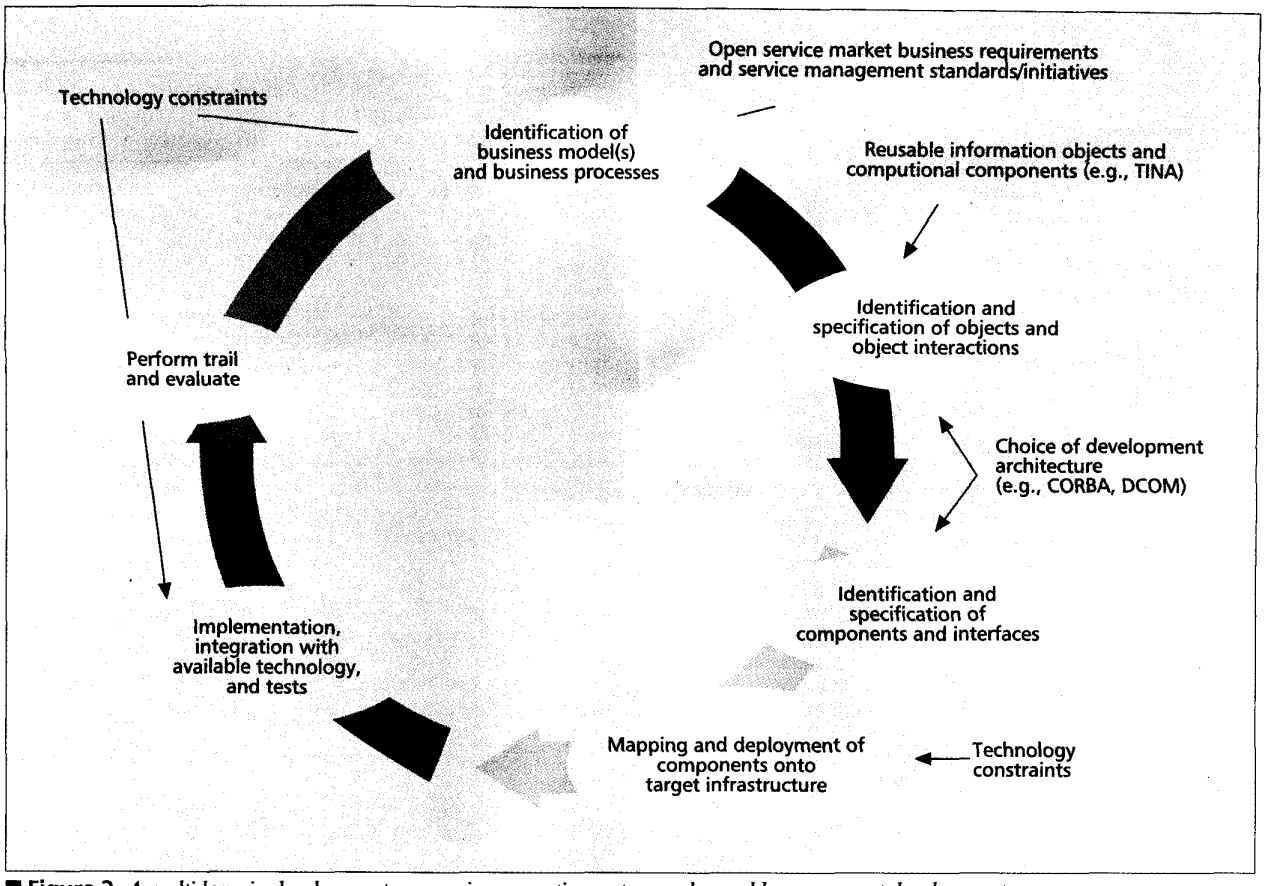
For this reason component agents were developed which interface the workflow engine to the components. The agent source code is over 50 percent generic because the interface to the workflow engine is standardized across all workflow agents, and only workflow control data is passed between the engine and agent(s). The management-component-specific part of the agent is responsible for retrieving the information required to invoke a management component. This information is stored in the shared (component) data server, the interface to which is again common for all agents. The agent is also responsible for placing any resultant information required to be shared into the shared data server. A *wrapper* object, either remote or running in the virtual memory space of the agent, actually interacts with the management component. This allows the component to operate in the environment for which it was originally designed.

The agent lets the workflow engine know that specific management activities have been completed by sending events or one-way asynchronous calls to the WIS, which has a number of registered receivers that need to be notified of such completions. These receivers include, but are not necessarily limited to, the scheduler and dispatcher. The use of asynchronous invocations (or events) between the agent, WIS, scheduler, and dispatcher allows a greater degree of concurrency, less chance of activity blocking, and more flexible integration of the workflow engine itself. The scheduler, dispatcher, and agents were all implemented in Java, running on Win NT. The WIS and management rule base used commercial database and knowledge base systems. CORBA (Orbix 3.0) was used as the distributed platform for the workflow engine.

Element #3: Representing the Business Process

The third vital element of successful integration of management systems is that the integration must be driven by specific business needs. Interoperability and cooperative systems are expensive to develop and complex to operate. Therefore, the business processes, which traverse management system boundaries, must be well described and depicted in technology-neutral ways. By maintaining a focus on management information and control flows which satisfy specific user requirements, the integrity of solutions developed spanning different organizational domains and technologies may be more readily assessed.

At the analysis stage in many OO design methodologies, use cases are a very popular way of analyzing the interactions between a system and the actors that use it at the system boundary. However, use cases are not good at describing the internal operation of a system. Even if subsystems are identified, and use cases for each subsystem generated, these use



■ **Figure 2.** *A multidomain development process incorporating system and reusable component development.*

cases may only interact with external roles, or subsystems modeled as roles. There is no well understood mechanism for tracing the interactions between different use cases in different subsystems.

However, the identification of interactions between subsystems is typically the kind of analysis performed in business process reengineering activities. Here, businesses aim to define the major processes they provide to their customers, which at a high level can be adequately captured with use cases. However, the aim of business process reengineering is to analyze the internal processes of an organization to understand how they interact to provide value to customers, and how the structure of these processes and their interactions can be changed to improve customer services and reduce costs. This problem is complicated for management systems by the interactions often required with processes in other organizations. The framework's development methodology must therefore address the problem of analyzing the requirements for multidomain management tasks that must be performed by interactions between management processes, some of which will support automated interactions and some of which will not.

A common representation of such control flow is an event-driven process chain, a graphical modeling technique that allows activities to be associated with organizational roles and with objects representing business information. As described in [8], the inclusion of activity diagrams in UML allows it to support a similar type of modeling diagram. The analysis of a specific management task in a specific business scenario can therefore be initially described in terms of a use case giving the interactions of the system with human roles. The use case may not necessarily mention the internal business processes. Therefore, these processes need to be analyzed using activity diagrams. The activities can be placed within "swim lanes" representing TMF business processes, possibly residing in different administrative domains. This will ease the identification of where existing TMF business agreements match the requirements of the task at hand.

Management Development Process Employing Reusable Components

This section proposes a customized management development process. The development process facilitates the design of both reusable management components and the overall multidomain systems development which is composed of reusable management components. The development process indicates where architectural and integration technologies influence the design of the components and systems. The development process generates the relevant information required by a workflow management system to perform automated, flexible component integration and interoperability.

The proposed multidomain management development process is an incremental process which supports the phases of requirement capture, analysis modeling, deign modeling, implementation, testing, deployment, and integration. Figure 2 presents the design process as a cycle. The main cycle (signified by the large circular arrows) identifies the actual steps that must be performed at each stage of the development of the management components and systems. The text external to the design process illustrates the influences and constraints particular to the development of management sys-

tems within the context of the target implementation environment.

The development process above exhibits a coexistence of OO development and component-based development methodologies. In the context of multidomain management systems, an OO development methodology is used to describe a set of business processes that will be supported by a number of reusable management components which will solve specific business problems.

A Development Process for Reusable Management Components

Overview — Reusable components are typically presented to system developers as sets of libraries, that is, a set of software modules and definitions of the individual operations of the component. Thus, a component is presented in terms of that component's design model and its software. This can cause problems if changes are required in order to reuse the component. Consider, for example, a component which is part of a framework. The framework may be general (e.g., CORBA services) or aimed at supporting systems solving problems in a particular problem's domain (e.g., the TINA service architecture). In either case the framework will provide (or assume!) some high-level architectural and technological guidance on how components can be integrated together and how they can support the development of the system. Such frameworks are often considered at the analysis stage, so the analysis model is structured in a way that accommodates the inclusion of the framework's components into the design model. However, frameworks typically only give general guidance on the use of components, and the suitability or otherwise of individual components in satisfying requirements still needs to be considered in the design process.

For telecommunication management systems development, such a typical component reuse situation is difficult to apply because there is no commonly accepted framework that supports a suitably wide range of components. Currently this is a problem domain where several different, though probably complementary, frameworks can be used, such as TMN and SMFs, CORBA, and TINA. The TMF aims at trying to define this framework, which encompasses these and other suitable architectural and technological frameworks; however, no such overarching framework is currently available.

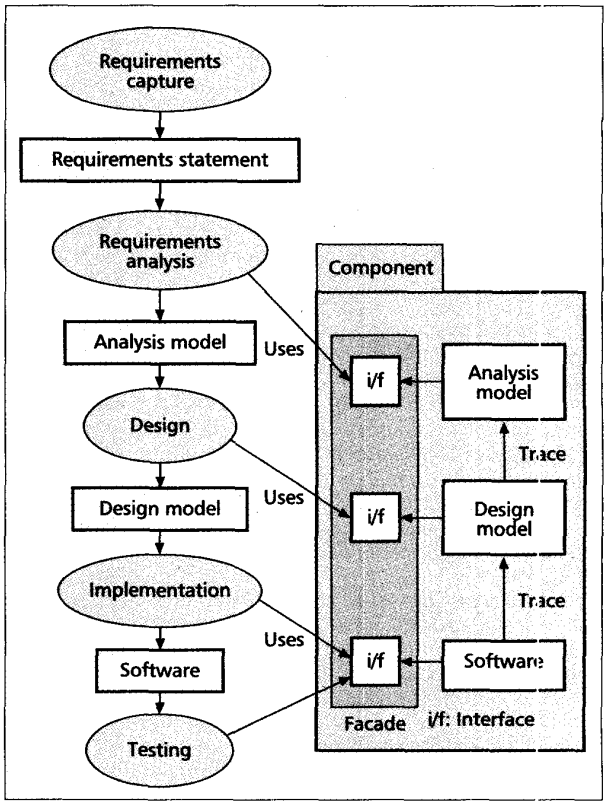
This development process for management component reuse is motivated by the absence of such a well-defined common framework. Instead, it attempts to provide guidance on how components can be specified in a more self-contained manner that is commonly understood. In this way, decisions about reuse can be made based on the suitability of individual components rather than a wider assessment of the suitability of an entire framework. The approach is aimed at making decisions based on the architectural and functional aspects of a component rather than its technology. The technology is treated as an orthogonal issue that is handled primarily through the employment of suitable integration technologies and techniques.

The approach proposed for reusable component development is based on Jacobson's OOSE. The basis of the approach is that components are not presented just as units of design and software, but should be packaged together with the analysis model of the component rather than strongly integrated into a specific component framework. If the modeling techniques used for the analysis model of the component are similar to those used for the modeling of the system in which they might be included, it is much easier to ensure that the component is suitable for use in the system. In addition, the system's analysis model can directly use the abstractions of the various components it reuses, easing the task of requirements analysis, and ensuring compatibility between components and system requirements at an early stage. This analysis-model-based reuse approach is depicted in Fig. 3.

The presentation of a component for reuse is known as a *facade*. A facade presents the reuser of a component with the information needed to effectively reuse the component, while at the same time hiding from the reuser unnecessary design and implementation details about the component. In the analysis-model-based reuse approach, the facade consists not just of reusable code and the design model details needed for reuse, but also the analysis model relevant to the design model part of the facade. The following section provides more details on the generation of facades.

Reuse with Facades — A component may present several different facades, possibly aimed at different types of reusers (e.g., black- or white-box reusers). A component may have various releases of a facade to reflect the evolution of the component. The usefulness of the facade is strengthened if there is clear traceability between the different models so that within the facade reusers can easily determine which parts are useful for them, working down from the analysis model level.

Obviously, the construction of a facade from a component's internal models, generated during its software development process, will be greatly eased if the same type of



■ **Figure 3.** *Reuse using component analysis models.*

modeling approach is used for this process. Strong traceability between the components internal models will ease the selection of parts of the model for inclusion in the facade, based, for instance, on a set of use cases. However, it is not a requirement for the component to have been originally developed and documented using an OOSE-like process, and part of the benefit of facades should be that it can hide the internal model if necessary.

In deciding what level of granularity to choose for identifying reusable components, the following must be considered:

- A component should provide some useful level of functionality to the reuser.
- The component should only have loose coupling to other components. If one component cannot operate in a useful way without the presence of another, consideration should be given to merging the two components into one.
- The more flexible a component, the more reusable it tends to be.

Concerning the latter point, the flexibility of a component can be captured using a variety of *variability mechanisms*. Some typical variability mechanisms that can be used within a facade are:

- *Inheritance:* This can be used to create subtypes or subclasses of abstract classes present in a facade.
- *Uses:* This is the mechanism for inheriting from an abstract use case.
- *Extensions:* These are small attachments used to express variations in the behavior of a use case or object.
- *Parameterization:* This is the use of templates, frames, or macros.

Other mechanisms which could be applied across a set of components or within a complex component are:

- Configuration languages used to connect optional or alternative components in complete configurations.
- Generation of derived components and their relationships from languages and templates. This is more the ultimate aim of ODP viewpoint languages, and languages such as SDL.

A full tutorial-style illustration of the design steps for component development is beyond the scope of this article due to article length restrictions. However, a worked tutorial-style example can be viewed at [9]. It consists of UML use case models, UML class and object models, and UML activity diagrams (which provide a mapping of which components are responsible for which activities).

Multidomain Management Development with Reusable Management Components

The proposed multidomain management development process is the modeling of the business processes and constituent activities, the mapping between these management activities and reusable management components, and the integration of the components with the workflow engine.

The steps involved in analyzing a business management problem are as follows:

- Identification of business model(s) and business process: The modeling of the business situation in question to identify business actors, the business roles they play, and the responsibilities the roles play with respect to each other; the modeling of the boundary of the multidomain management system in question and the required interactions of the system with external actors.
- Identification and representation of management process activities: involves modeling the business tasks as control and information flow between activities
- Specification of management process rules
- Identification of reusable components, specification of shared data requirements
- Mapping of management activities onto available (reusable) management components
- Development of (interface) agents (between workflow engine and management components)
- Testing and trialing of workflow-based multidomain management systems

Workflow-Based Integration of Reusable Management Components to Implement Specified Business Processes

Three of the steps of the multidomain management development process were specifically involved in the combination of the business processes, integration technologies, and reusable management components, namely:

- Specification of management process rules
- Identification of reusable components, specification of shared data requirements
- Development of (interface) agents (between workflow engine and management components)

These three steps are specific to the workflow component integration approach. The specification of management process rules is derived from the activity diagrams. The rules, however, can be more detailed and expressive. For example, conditional branching can be based on either process structural rules or values computed by management component execution and stored in the shared data server. The rules themselves are dependent on the rule-based technology used in the particular workflow engine. Example representations are expert system production rules, petri nets, or control statements stored in a database.

In order to support the specification of shared data requirements of the reusable component, the shared data server supports an object serialization and persistency services. The interface of the shared data server supports searching and retrieval of the shared data based on a management process instance. The component agents who perform the storage and retrieval of shared information use the unique keys of the process instance identifiers and activity identifiers to select the appropriate information. These keys are part of the workflow control information, which is originally generated by the workflow engine and passed between the workflow engine and the agents.

The development of the interface between the workflow engine and the management component is by necessity specific to each agent. However, the facade of the management component provides more than enough information to perform this development.

Summation

This article proposes three crucial elements of multidomain management system development. A rationalization as to the significance of each of these elements is presented, and an analysis of the current trends in management system development processes is described. However, in order to harness these elements, an integration of these elements is proposed and illustrated. The objective of the three elements discussed in this article is not to invent completely new development processes, integration techniques, and business processing representations, but rather to enhance and extend their application to support the development of multidomain management systems. The benefits of the approach described in this article are that management systems can be implemented from a heterogeneous

collection of management components (i.e., not from one framework, no requirement to be part of a single "creation" environment or a single CASE tool). The article combines component development techniques and a multidomain development process, which successfully harnesses the capabilities of a workflow engine for management system development. Such development processes would not be conceivable without such tools. In the development of demonstration systems (used in the illustrations presented in this article), Rational Rose and Paradigm Plus were used. Current implementation experiments are ongoing with fulfillment, assurance, and accounting multidomain business processes. Different integration technologies are also being experimented with to provide a comparison with the telecommunication management workflow engine.

Acknowledgments

The authors would like to acknowledge the European ACTS program which partially funded the research described in this article, and to thank the research teams of the ACTS Prospect (contract number AC052) and Flowthru (contract number AC005) projects for their cooperation and feedback.

References

[1] M. Fowler and K. Scott, *UML Distilled — Applying the Standard Object Modeling Language*, Prentice Hall, 1997.
[2] G. Booch, J. Rumbaugh, and I. Jacobson, UML Doc. Set 1.1, "Rational Rose," 1997.
[3] NMF Telecoms Operations Map, NMF GB910, stable draft 0.2b, Apr. 1998.
[4] A. Vincent and C. Hall, "Modeling/Design Methodology and Template," NMF internal doc., draft 4, Oct. 18, 1997.
[5] V. Wade *et al.*, *A Design Process for the Development of Multi Domain Service Management Systems, Guidelines for ATM deployment and interoperability*, S. Rao, Ed., Baltzer, pp. 88–103.
[6] N. Karatzas *et al.*, "TMN and TINA Coexistance," *Guidelines for the Interoperability of Broadband Networks*, P. van Binst, Ed., Baltzer, Jan. 1998.
[7] M. Rusinkiewicz and A. Helal, "Workflow Management Systems," *J. Intelligent Info. Sys.*, vol. 10, no. 2, Mar./Apr. 1998.
[8] T. Allweyer and P. Loos, "Process Orientation in UML through Integration of Event-Driven Process Chains," *Proc. UML '98*, Mulhouse, France, June 1998.
[9] http://www.cs.tcd.ie/vwade/ThreeKeysExample.html

Additional Reading

[1] F. Nesbitt, T. Counihan, and J. Hickie, "The EURESCOM P.610 Project: Providing a Framework, Architecture and Methodology for Multimedia Service Management," *Proc. 5th Int'l. Conf. Intelligence in Service and Networks*, Antwerp, Belgium, Springer-Verlag, 1998.

Biographies

VINCENT P. WADE (Vincent.Wade@cs.tcd.ie) is a lecturer in the Computer Science Department in Trinity College Dublin, Ireland, and has been researching the area of network-based services for over 12 years. He received his B.Sc. from University College Dublin, and an M.Sc. from Trinity College Dublin. He is head of telecommunication management research in Trinity College and is active in research on distributed management systems, broadband information services and virtual environments. He currently leads several EU and industrial research projects in these areas and is author of over 40 technical papers in international conference and research journals.

DAVID LEWIS (DLewis@cs.ucl.ac.uk) graduated in electronic engineering at the University of Southampton in 1987 and received an M.Sc. in computer science from University College London in 1990, since when he has worked as a research fellow. He has worked primarily on the EU-funded projects PREPARE and Prospect, in which he has been responsible for planning high-speed international testbed networks and leading teams developing integrated multidomain service management systems. He is also working on a Ph.D., researching a service management development architecture for the open services market.