

IMK2 - A database-centric, web service based architecture for distributed resource-oriented queries

Lavinio Cerquetti, Doan Isakov, Annett Priemel and Wolfgang Benn

Abstract—This paper will provide a summary of the IMK and IMK2 distributed software systems developed in the frame of the CRC457 research project. CRC457 is a research project of the Chemnitz University of Technology and deals with autonomous, elementary performance units, so-called competence units, which possess the ability of dynamically organizing themselves into non-hierarchical production networks at a regional level. IMK is a database of competence units which supports intelligent and partial multidimensional queries. First, the IMK software system will be described briefly and its structural limitations will be pointed out. Then, the motivations and the goals of the IMK2 reengineering project will be analyzed, with a particular focus on web service orientation, scalability, extensibility towards new content types and unitary access paradigm. Finally, the design and the structure of the IMK2 software system will be presented and discussed.

Index Terms— Competence Unit, Database Technology, Intelligent Multidimensional Query, J2EE, Regional Production Network, Software Architecture, Web Service Application

I. INTRODUCTION

The IMK and IMK2 distributed software systems have been developed in the frame of the CRC457 research project, whose goal has been to investigate the possibility for clusters of small autonomous competence units – the small and medium-sized enterprises which, holding a 50% share of the gross added value and almost two thirds of the employees in Germany, make up the spine of its economy – to effectively share their resources and competences by dynamically reorganizing themselves into customer-oriented and service-based non-hierarchical networks finalized to the production of complex goods [1, 2, 14].

The acronym IMK stands for *Informationstechnischer*

Manuscript received December 15, 2006. This work was supported by the Deutsche Forschungsgemeinschaft (DFG) as a part of CRC457.

Prof. W. Benn. Author is with the Department of Computer Science, Chemnitz University of Technology, Chemnitz, 09107 Germany (e-mail: wolfgang.benn@informatik.tu-chemnitz.de)

A. Priemel. Author is with the Department of Computer Science, Chemnitz University of Technology, Chemnitz, 09107 Germany (phone: +49 371 531 31521; e-mail: annett.priemel@informatik.tu-chemnitz.de).

L. Cerquetti. Author is with the Department of Computer Science, Chemnitz University of Technology, Chemnitz, 09107 Germany (e-mail: lavinio.cerquetti@informatik.tu-chemnitz.de).

D. Isakov. Author is with the Department of Computer Science, Chemnitz University of Technology, Chemnitz, 09107 Germany (e-mail: doan.isakov@informatik.tu-chemnitz.de).

Modellkern, which could be translated as “Core Information Model”. The Core Information Model is a database of competence units, which possess individual competences, have various resources at their disposal and occupy a location.

A. Competence Units

Competence units (see Figure 1) are highly adaptive elements and are capable of extending and altering their competences, as well as of passing them on and inheriting them from interconnected components. They can possess both technical and organizational responsibilities and are able to influence and to adapt to their environment dynamically.

Competence units build up non-hierarchical networks by connecting directly with each other and share equal rights in each decision making process.

Favored by the lack of a hierarchical structure, competence units exist independently and are in free competition with each other. Regional proximity allows them to group temporarily in order to achieve technological, economic, ecological or social benefits. A region can be defined as an economic area whose particular structure is shaped by local competences, such as mechanical engineering, microelectronics or mechatronics.

In almost every region it is possible to find a number of existing competence units in free competition with each other, which form temporary connections in the frame of a value-adding process chain with the purpose of satisfying customer needs. Lack of specific competences can be alleviated with the instantiation of new competence units or with the utilization of external ones.

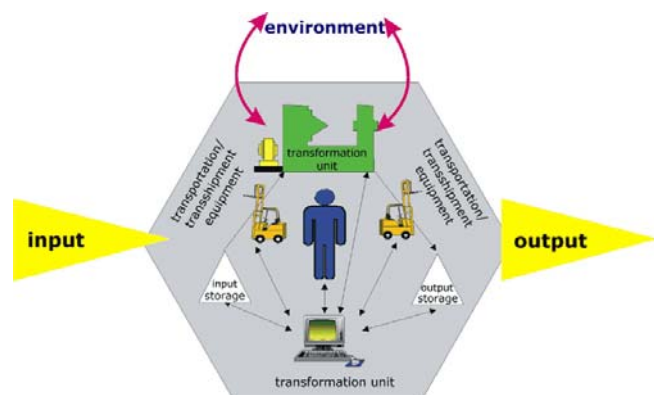


Figure 1. Competence Unit [18]

A categorization based on the notion of competence frameworks [17] is used to model separately the different steps

of a business process and of the value-adding process chain. A typical chain – as the one shown in Figure 2 - includes marketing and distribution, product development, work planning, manufacturing, assembly, logistics, quality control and service. Competence units originate from the synthesis of components from one or more competence frameworks. If the synthesis is carried out within a single competence framework, the resulting competence units will be function-oriented, whereas competence units which span the responsibilities of several frameworks are said to be process-oriented.

The following subsections will briefly discuss the competence unit types that are supported by the IMK and IMK2 software systems [13, 15, 16].

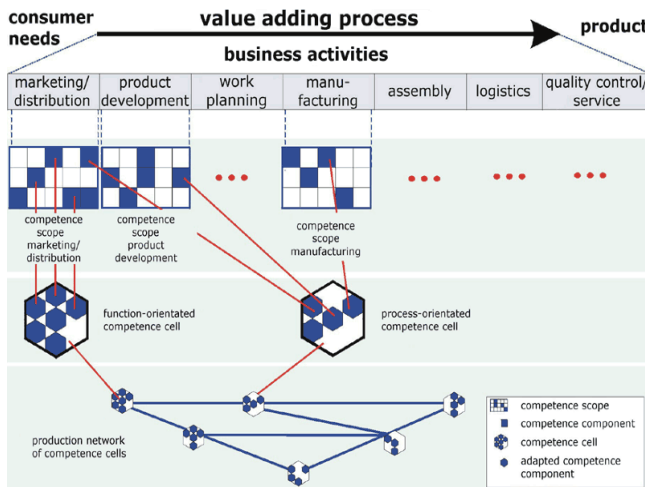


Figure 2. Value Adding Process Chain [18]

B. Structure of Competence Units

Figure 3 illustrates the general structure of a competence unit. In the scope of IMK, competences and capabilities of competence units are expressed with competence vectors and a so-called softfact matrix. A competence vector describes a specific action that can be carried out by a unit, and the competences of a unit are described by the logical OR of a set of competence vectors, whereas the role of a softfact matrix is to assess the capabilities of a competence unit in regard to their emotional state and their ability and readiness to cooperate with other units [12].

C. Logistician

A logistician is a competence unit which has several transport vehicles and typically one or more warehouses at their disposal. Their main activities are the transportation and storing of goods. The computation of transportation costs can be carried out either over third-party web services or locally, by using three different charging schemes based respectively on the volume or weight of the transported goods, or the length of the transportation route.

D. Manufacturer

A manufacturer is a competence unit that possesses one or more manufacturing machines, where each machine supports several assembly methods. A manufacturer is able to produce goods from raw parts or can assembly specific component parts depending on the assembly methods supported by their machines.

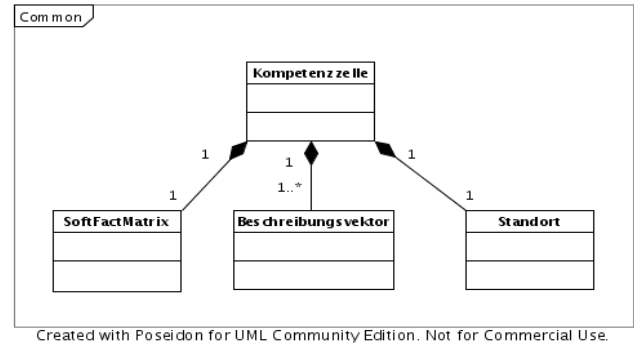


Figure 3. Competence Unit Structure in IMK

E. Production Planner

The task of a production planner is to create workflow schedules for the production of parts with varying size, weight, material and components. This competence unit has no resources at their disposal, and the level of its production planning competences is determined by an evaluation scheme based on the assessment of planning tasks.

F. Product Developer

A product developer is responsible for the development of new and the enhancement of existing products. This unit has a number of competence fields and is knowledgeable in product development methods. They have machine tools and assembly units at their disposal, as well as a number of additional resources like CAD software or patent catalogues, which are intended as aid in the product development process.

G. Quality Manager

Quality management encompasses the creation and execution of test plans, and involves two different types of competence units: test planners, responsible for the creation of test plans based on services and machine tools they are competent in, and measurement engineers, responsible for the actual execution of tests. Measurement engineers have access to special test tools, each of which supports a number of test methods and can be carried out on one or more materials.

II. STRUCTURE OF IMK

The competence unit roles listed in the previous section have been used as a heading point in the design of IMK. Every role has been analyzed, modeled and implemented independently. Therefore, the IMK project consists of five separate subprojects that are only slightly interdependent, but which exhibit common structural patterns. Each of the subprojects can be

viewed as a two-tier software system, consisting of a database layer and an application logic layer, developed in a mix of Java, C# and C++.

Furthermore, selected IMK functionalities have been made accessible by way of SOAP [3] web services. It should be mentioned that the implementation of functionally complete user front-ends - whose availability would extend the IMK project into a workflow management system - was not one of the goals of the project.

In order to maintain a natural representation of information at all system levels, a pure object-oriented database has been used as data repository. The existence of independent design models has concretized itself into a proliferation of inconsistent and partially redundant database schemata.

Each subproject exports a basic set of reading and writing functionalities by way of SOAP web services. Additionally, the logistician and quality management subprojects expose web services which allow to load and unload static data in batch mode, as well as to perform native queries in OQL [4], the object-oriented query language partially supported by the underlying database engine. Unfortunately, due to separate modeling phases, the IMK web services do not comply to any standard and are syntactically and semantically inconsistent.

For the production planner and manufacturer subprojects, client-server graphical front-ends have been developed. These "fat" clients are tightly coupled with the persistent object schema and do not make use of the web services, as they need extended functionalities which are not exposed by the IMK application layer, such as loading and unloading of structural data and content validation.

III. LIMITATIONS OF THE IMK IMPLEMENTATION

The IMK software system suffers from a number of limitations, most of them due to the separate development process used for each subproject and to the technological choices made in the design and implementation phases. These limitations, which will be described in the following paragraphs, have led to the decision to remodel and reengineer the IMK software system, and have given birth to the IMK2 project.

First, the IMK system as a whole does not have a unified semantic model and a homogeneous class hierarchy, which results in a number of inconsistent behavioral patterns and partially redundant class definitions. Each subproject takes a different approach in representing the structural details of competence units and does not necessarily comply with the structure described in 1.1. These inconsistencies severely limit the extensibility and the interoperability of the IMK project and greatly reduce its robustness and solidity.

Second, the choice of basing the implementation of IMK on direct database access rather than on a proper component and distribution framework has placed an enormous structural burden on the application developers and has, as a matter of fact, resulted in an extreme lack of software modularization and

in the absence of any distribution feature within IMK, making it unsuited to handling large volumes of data and traffic.

Third, web services have been implemented in IMK as an afterthought. The core system structure is strictly client-server, and the amount of functionalities exported by the SOAP web services is extremely limited and absolutely unsuited to the implementation of fully functional application clients.

Fourth, partitioning the system into five independent subprojects has led to evident redundancies in the application logic and in the database layers of the IMK system. Persistent classes are duplicated in the application logic layer and consequently in the database schemata as well.

Fifth, using a pure object-oriented database as data repository has made any kind of consistency check extremely hard, as all integrity rules are completely offloaded to the application developers. This causes the proliferation of invalid data and leads once again to redundancies and inter-project inconsistencies.

In fact, a number of inconsistent database states – often consisting in uncontrolled object duplication - have been detected during the everyday use of IMK. Generally speaking, these problems arise from having the object-oriented database assign an identity to every newly created object, even if that object has already been stored and only needs to be resolved. The task of object resolution is not trivial for complex object hierarchies and is once again left to the application developer, resulting in an additional burden on the application code.

Finally, the implementation of the Object Query Language (OQL) supported by the database engine chosen for the IMK project presents a number of important restrictions, which have led to incomplete query specifications and structural performance penalties. Consequently, additional filtering of data has to be carried out in the application logic layer, resulting in a sharp data traffic increase and longer execution times.

This situation is aggravated by the fact that partial object tree reconstruction is not supported by the object-oriented database engine, thus making navigation of non-trivial data extremely slow and inefficient. Figure 4 is an attempt to illustrate the problem at hand. Assuming that an instance of Class 1 associated with a given Root instance has to be retrieved from the database, the database should only deliver the Root object and the corresponding Class 1 object. However, the lack of partial object tree reconstruction makes it impossible to retrieve only those two objects, thus the whole object hierarchy has to be reconstructed in memory in order to access the specific Class 1 instance. Accompanied with an increase of the depth of class hierarchy, this feature severely limits the scalability of the whole system.

IV. DESIGN GOALS OF IMK2

The decision to remodel and reengineer the IMK project has arisen due to its structural limitations and with the goal of increasing its interoperability, maintainability, efficiency and

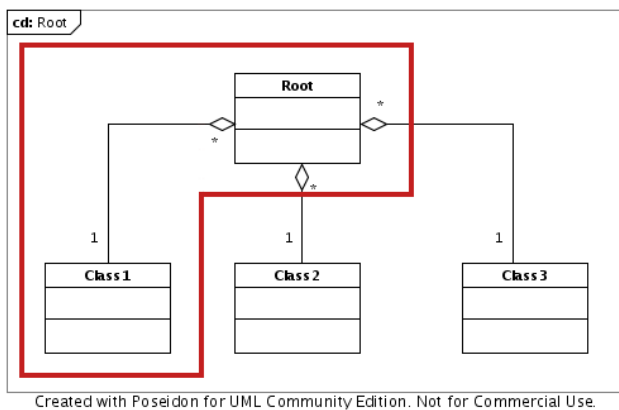


Figure 4. Partial Object Tree Reconstruction in IMK

extensibility, as well as to improve the integrity and consistency of the stored data. One of the focal points has been switching from the original two-tier architecture to a scalable multi-tier software system.

The most severe limitations in IMK resulted from the lack of a unified semantic model and a harmonic class hierarchy. The existence of five independent analysis and design models has led to five conflicting interpretations and implementations of the main concepts and ideas, which have made extending the project a burdensome task. Thanks to the unification of the five semantic models, most of the redundancies and inconsistencies both at the application and at the database layer have been solved, resulting in a single database schema and in a coherent class hierarchy. The separate model sets which characterized each functional area have been merged and rearranged entirely. Semantically identical classes have been consolidated, up to member definitions and method signatures, and all competence units have been modeled on the basis of a common interface that specifies clearly the unit structure. As a matter of fact, all competence units in IMK2 follow the structure described in subsection I.A.

Although the web services implemented in IMK exposed only a limited set of functionalities, the recognition of their central role in the structure of a multi-tier software system has led to the decision to remodel IMK around them, turning them from an optional commodity into the central access point to all IMK features. In fact, all IMK2 functionalities are exposed as platform independent SOAP web services. To maximize platform neutrality, all input and output information is represented as document-based XML data, which frees the front-end developer from the need of having to define custom XML mappings for application specific data types.

One additional goal which was pursued from the beginning in the remodeling and reengineering process was backward compatibility. As the IMK software system was already in use and changing the existing interfaces would have broken all the existing front-ends, a decision was taken to keep intact all the old web service signatures and to add new ones, which comply with a unified standard and offer an extended set of functionalities. The old web services, although still available,

have been deprecated and new frontends are encouraged to make use of the extended and standardized web services provided by IMK2.

Furthermore, reorganizing IMK2 as a web service and XML based distributed application has made it possible to switch to an infrastructure based on thin software clients, freeing the front-ends from all knowledge about the structure and the behavioral characteristics of the back-end data objects, whereas in the original IMK project clients were strictly coupled with the data objects on the server side, which resulted in the need to recompile and update all client installations whenever the data schema or the business processes had to change, with the risk of having obsolete applications performing inconsistent database updates. Another area in which IMK was clearly lacking was scalability. The choice of granting business objects an unmediated access to the data repository and not having any middleware taking care of component allocation and transaction synchronization made the IMK architecture seemingly easy, but in fact placed an excessive burden on the developer, leading to a poor software modularization, a heavy bottleneck in the database access, a superficial transaction management and to non-existent scalability and distribution strategies. In order to solve these issues, the architecture of IMK2 has been centered around J2EE [5], a well-known distribution, resource management and componentization framework.

As keeping the object-oriented data repository in a consistent state proved to be exceedingly difficult, it was decided to migrate the data tier to a mixed object-relational architecture, which would provide a better ground for validation and integrity rules and deliver a performance advantage. Furthermore, it was decided to merge the separate object schemata developed for IMK into a single and coherent data schema. A unified semantic model was a prerequisite for this choice and was needed in order to achieve a real database consolidation rather than a simple inclusion of all pre-existing relations into a single database schema.

Finally, the usage of EJB3 [6] - a state-of-the-art J2EE technology - has been the cornerstone in removing all database dependencies from the application and middleware tiers. The former is strictly based on a document-oriented XML data representation, whereas the latter manages the data in a purely object-oriented way. Although at the time of development the EJB3 specification had not been finalized yet and only partial implementations were available, the evident advantages associated to this technology have motivated its integration into IMK2, whose middleware tier makes wide use of the declarative support for object-relational (OR) mapping delivered by EJB3 POJO objects.

Furthermore, the availability within the J2EE framework of EJBQL [7], a semantically rich and object-oriented query language, has removed the restrictions imposed by the limited capabilities of the OQL language supplied with the object-oriented engine used in IMK. Thanks to this, all query logic has been respecified in a completely declarative way and the extensive object navigation required in IMK has become

unnecessary.

V. MODEL OF IMK2

The model of IMK2 is based on a unified view of competence units (see Figure 5), represented by instances of the class *Kompetenzzeile*. This container class describes the general structure of all competence units and serves as an aggregator for so-called role classes, which model the specific tasks and responsibilities of each concrete competence unit type in the production process.

While most instances of *Kompetenzzeile* will usually be defined in terms of a single role component, the IMK2 model does not inherently limit the number of roles which can be associated to a single competence unit, allowing for the existence of complex, multi-function competence units which can concurrently perform different work assignments in separate production chains.

Each role aggregates one or more competence vectors (class *Beschreibungsvektor*), which comply to a standard interface in order to ease the handling of structural information throughout all the application layers. Hence, the IMK2 model contains a generic interface and reusable algorithmic classes for the competence vector types. A major advantage of this approach is that the developer is freed from the error-prone task of implementing a consistent behavior among all competence vector classes.

The competence vectors play a central role in completing the global model of unified competence units. They aggregate all resources and capabilities and are the bridge element between the model described earlier and the characteristics of the specific competence unit roles. The following subsections will clarify how the model was extended with the functional area of each competence unit.

A. Production Planner

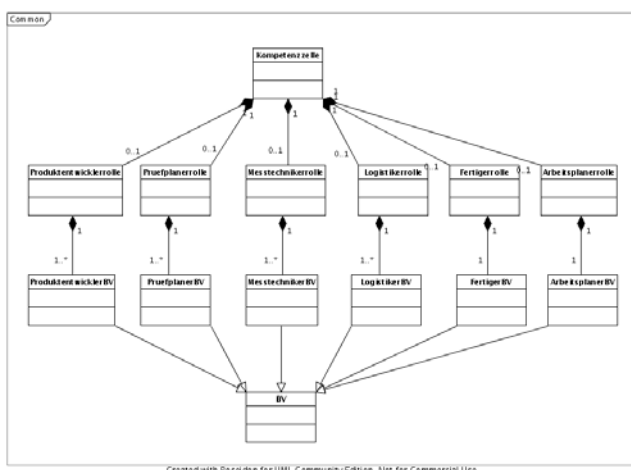


Figure 5. Competence Unit Structure in IMK2

The production planner competence vector aggregates production planning offers (class *Angebot*) made for single business objects (class *Geschäftsobjekt*). In the context of

IMK2, a business object is mostly a concrete component and can be produced with different sizes (class *Teilraum*) and weights (class *Gewicht*). The instances of these two classes are aggregated in the offer class and prevent duplicated business objects with differing size and weight from appearing in the database.

Furthermore, an offer aggregates one or more material types (class *Werkstofftyp*) in which the product can be delivered, and at least one form type (class *Teiltyp*), which determines the form of the produced object. A production planner can make use of a number of production planning skills (class *Arbeitsplanungskompetenz*) in order to deliver the final result. Planning skills can have a variety of characteristics depending on their type and are represented with an inheritance tree of depth two with the class *Arbeitsplanungskompetenz* being the root of the tree. Moreover, planning skills can extend each other, imposing the self aggregation of the root.

The planning skills of every production planner are evaluated at different time points and their ratings are saved in instances of the class *Bewertung*. Evaluated planning skills (class *BewerteteKompetenz*) are aggregated in the offer class and serve as a guide for further planning.

A standard query for production planners specifies the name of a single business object (<geschäftsobjekt>) for which a production plan needs to be derived, its desired weight (<gewicht>), size (<hoehe>,<breite>,<tiefe>) and form (<teilklass>). Furthermore, the query has to include the material (<werkstofftyp>) for the business object and a minimum rating (<bewertung>) for the production planning skills (<aplkpz>) required for the fulfillment of the task.

B. Product Developer

The competence vectors of product developers aggregate their specialization field (class *Fachbereich*), the methods needed in the product development (class *Methode*), assembly units (class *Baugruppe*), machine tools (class *Werkzeugmaschine*) and non-personal resources (class *NPRessource*). All classes, with the only exception of *Fachbereich*, are self aggregated and build hierarchies of the corresponding instances.

The product developer query specifies a number of profiles (<profil>) containing required specialization fields (<fachbereich>), methods (<methode>), machine tools (<werkzeugmaschine>), resources (<ressource>) and a small number of further characteristics (<charakteristika>) that are contained in each competence vector. Each of the elements mentioned above can be omitted inside a profile if it is not relevant. A competence unit fulfilling the requirements possesses competence vectors which correspond to all profiles defined in the query, i.e. all profiles have to be satisfied concurrently.

C. Manufacturer

The manufacturer competence vector aggregates machines (class Maschine). A machine instance is connected with a workpiece support instance (class Werkstueckaufnahme) and a chuck instance (class Werkzeugaufnahme) building the core of the manufacturer resources, and it is associated with a machine type (class Maschinentyp). Furthermore, it has a size and area of effectiveness specified as instances of the class Teilraum and references an interface (class Schnittstelle) which determines its interoperability.

Both chuck instances and workpiece support instances are associated with a corresponding type. Those determine mainly the interfaces, coupling joints (class Trennstelle) and resilience types (class Spannkraftart) supported by a concrete realization. Additionally, they specify whether the realization can be modularly built and extended, and whether it is mobile. The chuck and the workpiece support classes aggregate three Teilraum instances which respectively determine their size as well as the maximum and minimum size of the processed workpieces and tools. Both of them are associated with a single instance determining their resilience type.

A chuck type is associated with at least one tool type (class Werkzeugtyp), determining the tools that can be combined with a particular chuck instance. A tool type references a number of coupling joints, specifying a set of joints for a specific tool instance (class Werkzeug). A chuck aggregates at least one tool instance with a suiting coupling joint.

The processing of a manufacturer query is based on a production plan, an example of which is shown in Figure 6. A production plan specifies mainly an assembly unit (<einzelteil>) to be produced, the raw materials (<rohteil>) that are going to be used in the production process and the manufacturing methods (<fertigungsmethode>) that need to be applied to these materials. The production is partitioned into separate manufacturing chains (<prozesskette>), where each chain is dedicated to the processing of a specific raw material. The manufacturing chains are divided into manufacturing steps (<prozessschritt>), where each manufacturing step is associated with a specific manufacturing method. Manufacturing steps are further divided into modules (<bearbeitungselement>) describing the specific manipulation sequence of a raw material. The aim of the query resolution is to find suitable competence units which can provide a machine able to carry out the specific task for every module. Comparisons are based mainly on the manufacturing methods associated with the machine types and the material types (<werkstofftyp>) associated with the tools in the chuck referenced by a machine.

D. Test Planner

The test planner competence vector aggregates activities (class Aktivitaet), services (class Dienstleistung), component part categories (class Einzelteilkategorie) and machine tools (class Werkzeugmaschine). Furthermore, the classes

Dienstleistung, Werkzeugmaschine and Einzelteilkategorie are self aggregated and build a hierarchical tree structure, in which every parent node extends the features of its children, hence leading to a query processing mechanism based on partial tree matching. A similar structure is used for assembly units (class Baugruppe), which are associated with a number of Werkzeugmaschine instances, specifying which units can be manipulated by a specific machine.

Furthermore, the test planner competence vector owns a number of instances of a weight class (class Gewichtbedingung), specifying the weight of the assembly



Figure 6. Excerpt from a production plan

units (class Baugruppe) that are aggregated by the Werkzeugmaschine instances. Component part categories are composed of a number of component parts (class Einzelteil), which inherit from the same part class (class Teil) as assembly units. The material of each part is defined by a single material instance (class Werkstoff). Services aggregate a number of inspection features (class Pruefmerkmal), determining what kind of tests a test planner can schedule on a given set of machines and units.

Test planner queries specify a number of activities (<aktivitaet>), services (<dienstleistung>) and inspection features (<pruefmerkmal>) that the competence unit must be able to carry out. Additionally, they either designate a number of component part categories (<einzelteilkategorie>) or explicitly list component parts (<einzelteil>), assembly units (<baugruppe>) or machine tools (<werkzeugmaschine>), i.e. only one of these elements can be specified. Every query should contain a type tag (<typ>), determining the elements which must be specified and those which are not allowed. The competence units which are returned after the execution of test planner queries own at least one competence vector, which aggregates the mandatory components of the query and supports a test plan for the specified materials or component parts.

E. Measurement Engineer

The measurement engineer competence vector is composed of a test tool (class Pruefmittel) defined by its area of effectiveness (class Teilraum) and associated to a number of test method offers (class PVAngebot), which reference a test method (class Pruefverfahren) and a material (class Werkstoff) the test can be performed on. Furthermore, a test tool specifies lower and upper bounds for the desired output values (class Sollwert).

Sollwert instances reference a test feature (class Pruefmerkmal), a test characteristic (class Pruefeigenschaft) and a group of characteristics (class Pruefeigenschaftsgruppe) determining the specific aspect described by the output value. The test characteristic instance must be included within the characteristics aggregated by the class Pruefeigenschaftsgruppe.

The execution of a measurement engineer query returns the set of competence units which are able to carry out a number of specific test steps (<prozessschritte>). A sample measurement engineer query with a single test step is shown in Figure 7. The processed element (<bearbeitungselement>), its material (<werkstoff>), size (<hoehe>,<breite>,<tiefe>) and weight (<gewicht>) have to be declared in each test step. Furthermore, a test step includes a test method (<pruefverfahren>), which is going to be applied to the processed element, features which are going to be tested (<pruefmerkmal>,<pruefeigenschaftsgruppe>,<pruefeigenschaft>) and the desired output values (<untererToleranzwert>,<obererToleranzwert>).

F. Logistician

A logistician is responsible for the transport and temporary storage of parts and assembled products. Its competence vector describes an optional set of warehouses (class Lager) and its vehicle fleet (class Transportmittel), each vehicle being defined by its speed, size and maximum capacity.

A logistician query is based on a set of deadlines and on a production step, which implicitly defines a starting (<quelle>) and a destination location (<ziel>), as well as a number of goods (<ladeeinheit>), and delivers the best offers among all



Figure 7. Measurement Engineer Query

the logisticians able to fulfill the transportation task, either in a single step or by way of intermediary storage. The computation of the charging schemes can be either performed locally - based on the static price data associated with the logistician role - or integrally offloaded to remote web services provided by each logistician.

VI. ARCHITECTURE OF IMK2

IMK2 has been designed around a unitary semantic model describing the competence units and a coherent set of J2EE and Java based platform specific models from which the system architecture and all software artifacts have been derived. Having a homogeneous class hierarchy has made it possible to implement a unitary software system, minimizing the inconsistencies and guaranteeing a consistent view over all data.

The architecture of IMK2 (see Figure 8) follows the current best practices of software engineering and is heavily based on design patterns [11]. In particular, IMK2 has been built on the principle of concern separation [9] and is organized as a multi-tier software system. In particular, four tiers can be identified and will be discussed in the following subsections: the client tier, the session tier, the business logic tier and the data tier. Tier transitivity is guaranteed at all levels.

The session tier and the business logic tier are hosted within the server layer, which has been implemented as a J2EE 1.4 distributed application and is based on the EJB3 component architecture.

A. The Client Tier

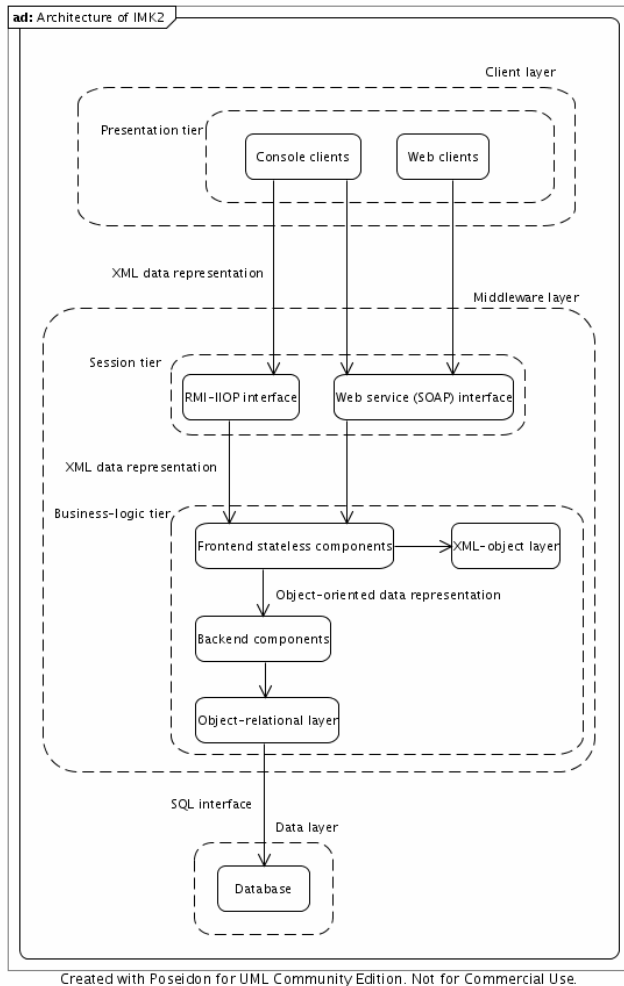


Figure 8. Architecture of IMK2

The client tier consists of user front-ends, i.e. software applications which have the purpose of exposing system functionalities to users.

Currently, IMK2 functionalities can be accessed by way of console and web clients. The former ones are meant to be used by system administrators and allow the user full access to IMK2 features. Console clients can interact with the middleware layer of IMK2 either with platform-neutral SOAP web services or using the RMI-IIOP [8] call protocol.

Web clients, which access the business tier over the IMK2 SOAP web services, expose only a limited subset of the IMK2 functionalities and are meant for general user access. Thanks to the backward compatibility of IMK2, a full reuse of the web clients developed within the original IMK project has been possible.

It should be noted that all information exchange between

client and business tier is based on XML data representation. This means that the choice of a particular access protocol on the client side has no impact on the user front-end logic.

Furthermore, as IMK2 strictly focuses on document-based XML schemata, no user-defined mapping is required and platform independence is maximized.

B. The Session Tier

The software components in the session tier allow access to the IMK2 functionalities hosted on the server side. All IMK2 features are exposed both over the RMI-IIOP protocol and over platform-neutral SOAP web services.

The explicit existence of the session tier is motivated by technical limitations in the J2EE software stacks available at the implementation time of the IMK2 project. As soon as full J2EE semantics are supported, all software artifacts which make up this tier will be derived and managed automatically by the J2EE middleware on the basis of the interfaces contained in the business-logic tier.

C. The Business-logic Tier

IMK2 semantics are implemented in the business-logic tier, which can be considered as the focal point of any multitier software system.

In order to increase the robustness and the extensibility of the IMK2 project, its business-logic tier is structured as a multi-tier system in its own.

The front-end to the business-logic tier is represented by a collection of stateless EJB3 session components, which encapsulate the access to all IMK2 functionalities and make use of an XML-OO mapping in order to deliver the object-oriented view on data which is required by the lower layers in the business-logic tier.

The front-end layer delegates its work to a set of backend objects, consisting of software components capturing business and process semantics based on a strict object-oriented data view. This middle layer is implemented as a collection of EJB3 session components and is based on the data access object pattern [10].

The lowest layer in the business logic tier is represented by the components delivering the object-relational mapping, which have been implemented as EJB3 POJO entity components. All business calls are ultimately mapped into a sequence of operations on data objects, which are then translated into concrete database calls.

Using the latest draft of the EJB3 specification has made it possible to express the OR-mapping in a declarative way and to keep the application code independent from the relational schema of the data.

Within the business-logic tier, all data access is based on object semantics, and all data extraction happens by way of EJBQL, the object-oriented query language defined by the J2EE 1.4 framework.

It should be noted that the powerful semantics and the increased efficiency delivered by EJBQL over OQL have made

it possible to remove the need for data navigation at code level, as well as to process every query without the need for additional filtering logic.

D. The Data Tier

The data tier is found at the bottom of the architecture of IMK2 and is concretely represented by an object-relational database engine which hosts the data repository.

All access to the database is mediated by the J2EE middleware, which converts each EJBQL query and object access into a corresponding set of native SQL calls.

As a matter of fact, the IMK2 software system is SQL-free and database-neutral.

VII. CONCLUSION

This paper has discussed IMK and IMK2, two software systems developed in the frame of the CRC457 research project.

After a short overview of the goals of the CRC457 project, the structure of IMK has been discussed together with its limitations, which have led to its remodeling and reengineering into IMK2, a distributed software system based on web services, whose main goals and advantages are:

- A unified semantic and data model.
- A unified class hierarchy, characterized by a single competence unit interface and with no redundant or overlapping class definition.
- A unified and consistent database schema.
- A distributed, scalable and multi-tier software system centered around platform-neutral and data independent web services.
- The usage of document-based XML representation for all information exchange to and from the IMK2 software system.
- A thin client layer: front-ends do not possess any knowledge about the structure and the business processes associated with the back-end data objects.
- Transparent distribution and allocation of data objects.
- Removal of object navigation in the application layer.
- Complete backward compatibility with the original IMK specification.
- Vastly increased efficiency, maintainability, robustness and extensibility.

As a matter of fact, the remodeling and reengineering performed in the frame of the IMK2 project have led to a significant increase in the system features while at the same time dropping by more than 50% the code size and the number of model classes.

REFERENCES

- [1] Institut für Mittelstandsforschung: Mittelstand – Definition und Schlüsselzahlen. Bonn, 21 December 2004. URL: <http://www.ifm-bonn.org/dienste/daten.htm>.
- [2] Bundesministerium für Wirtschaft: Kleine und mittlere Unternehmen. Früherkennung von Chancen und Risiken. Arbeitsheft, Berlin, 1998.
- [3] SOAP Version 1.2. W3C, 24 June 2003 URL: <http://www.w3.org/TR/soap12>.
- [4] OQL User Manual. ODMG, February 1998. URL: <http://www.cis.upenn.edu/cis550/oql.pdf>.
- [5] J2EE 1.4 Specification. Sun Microsystems, 24 November 2003. URL: <http://java.sun.com/j2ee/1.4/download.html#platformspec>.
- [6] EJB3 Specification. Sun Microsystems, 8 May 2006. URL: <http://jcp.org/aboutJava/communityprocess/final/jsr220/>.
- [7] Enterprise Java Beans Query Language. URL: http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/EJBQL.html.
- [8] RMI over IIOP. Sun Microsystems. URL: <http://java.sun.com/products/rmi-iiop/>.
- [9] Design Pattern: Separation of Concerns, 2005. URL: http://www.geometryit.com/tip/separation_of_concerns.pdf.
- [10] Design Pattern: Data Access Object Pattern, 2001. URL: <http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html>.
- [11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.
- [12] O. Görlitz. *Inhaltsorientierte Indexierung auf Basis künstlicher neuronaler Netze*. Shaker, 2005.
- [13] O. Görlitz, R. Neubert, and J. Mehnert. Anbietersuche für den Aufbau kompetenzzellenbasierter Fertigungsnetze, in: Tagungsband Vernetzt planen und produzieren – VPP2001. Chemnitz University of Technology, 2001.
- [14] E. Müller. Hierarchielose regionale Produktionsnetze. Theorien, Modelle, Methoden und Instrumentarien. Arbeits- und Ergebnisbericht 2003, 2004, 2005 zum Sonderforschungsbereich 457, Chemnitz University of Technology, 2005.
- [15] R. Neubert, O. Görlitz, and T. Teich. Abgleich von Angebots- und Anforderungsvektoren von Kompetenzzellen für die automatische Generierung von Prozessketten, in Journal: PPS-Management 1/2002. Berlin: GITO Verlag.
- [16] M. Rentzsch. Anbietersuche für Arbeitsplanungsaufgaben im Informationstechnischen Modellkern. Chemnitz University of Technology, 2004.
- [17] J. Mehnert. Gestaltung und Integration von Arbeitsplanungskompetenzen für hierarchielose Produktionsnetze. Chemnitz University of Technology, 2004.
- [18] Collaborative Research Center (CRC) 457. TU Chemnitz. URL: <http://www.tu-chemnitz.de/sfb457/en/>