

Dynamic Business Network Process Management in Instant Virtual Enterprises*

Paul Grefen^a, Nikolay Mehandjiev^b, Giorgos Kouvas^c,
Georg Weichhart^d, Rik Eshuis^a

^a*Eindhoven University of Technology, Netherlands*

^b*University of Manchester, UK*

^c*Exodus SA, Greece* ^d*Profactor GmbH, Austria*

Abstract

Nowadays, business supply chains for the production of complex products or services are likely to involve a number of autonomous organizations. The competitive market requires that these supply chains are highly agile, effective and efficient. Agility and effectiveness are obtained by forming highly dynamic virtual enterprises within supplier networks. We call these instant virtual enterprises (IVEs). The required efficiency of creating and operating IVEs can only be obtained by automated support for design, setup and enactment of business processes within these IVEs. This process support involves the dynamic composition of local processes of network members into global processes at the IVE level. This functionality goes significantly beyond traditional approaches for interorganizational workflow management. The approach, architecture and technology required for this dynamic network process management in IVEs are outlined in this paper. We show how the developed approach is applied in the automotive industry in the context of the CrossWork IST project.

* The work presented in this paper is part of the CrossWork project, supported by the European Commission in the context of the IST 6th Framework, Contract No. 507590.

TABLE OF CONTENTS

1	Introduction	3
2	The state of the art	5
2.1	Interorganizational workflow management	5
2.2	Service-oriented computing	6
2.3	Agent-based approaches	6
3	Dynamic business network processes in instant VEs.....	8
3.1	The BNP concept	8
3.2	The IVE and DBNPM concepts	9
3.3	System requirements for DBNPM in IVEs.....	10
4	Designing the system architecture	12
4.1	Starting with a separation of concerns	12
4.2	Refinement step one: detailing functionality	13
4.3	Refinement step two: adding knowledge	15
5	Implementing the system components	18
5.1	Choosing the software platforms	18
5.2	Goal decomposition and team formation	18
5.3	Workflow composition, verification and prototyping.....	20
5.4	Workflow enactment.....	20
5.5	Ontology and user interface support.....	22
6	CrossWork: putting the approach to the test.....	25
6.1	Developments in the automotive industry	25
6.2	The CrossWork case study	26
6.3	Implementation of the prototype system.....	29
7	Conclusions and outlook.....	32
8	References	33

1 INTRODUCTION

In the modern economy, we see the development of ever more complex products. This holds both for physical products as for non-physical services. In physical production, a good example can be found in the automotive domain. Here, we see that the complexity of automobiles has increased in a dramatic way: the inclusion of new features like safety systems, driver guidance and support systems, mobile entertainment systems, and climate control systems has increased the number of components in an average car significantly [Max04]. In the service industry, we see comparable increase of complexity of products, for example in the financial industry (where we see more and more complex products consisting of mortgage, loan, saving, and investment elements), in the world of healthcare, where more complex medical procedures and packages are offered, or in the telecommunications domain, where we see complex packages of wired and wireless subscriptions, internet access, etcetera.

Apart from the increasing complexity of products, we also see that the product life cycles of become shorter and shorter: new versions or generations of products appear in an ever faster pace – both fueled by fast technological developments and by increasing competitive forces on the international market. Where car models could live for many years in the past, nowadays they are replaced every few years. New financial products or new combinations of them appear in a continuous stream – and the same holds for telecommunication and healthcare services.

The complexity of products requires that the organizations that produce them need to collaborate in production supply chains or business service networks, where each partner in a chain or network contributes to part of the complex product: no single organization can produce the complex product by itself [Cor02]. This means that we see the emergence of virtual enterprises centered on the realization of specific products or classes of products and consisting of a possibly large number of autonomous organizations. The shortened life cycle of products makes that these virtual enterprises need to have a dynamic character: they are formed for new products and must be dismantled when products are abandoned again. To stay competitive in modern markets, the creation of dynamic virtual enterprises must be performed swiftly: where this might have taken at least several months in the past, the time frame must now be reduced dramatically, for example to a few days or even less. To realize this, we have to think of a new form of virtual enterprise, which we call Instant Virtual Enterprise (IVE).

The creation and operation of an IVE requires automated support to guarantee required levels of efficiency – both in terms of time and costs. Support for creation of IVEs includes automated tools that can determine appropriate members for an IVE, systems that can integrate the support of these members and systems, and tools that can map integrated business specifications onto existing information processing infrastructures in the IVE. Support for operation of an IVE includes systems for business process enactment and monitoring at both the global IVE level and the level of the individual member organizations in the IVE [Gr06b]. Creation and management of these dynamically created networks of global and local business processes is called dynamic business network process management (DBNPM).

This paper describes an approach towards dynamic business network process management (DBNPM) in the context of instant virtual enterprises (IVEs), focusing on the required concepts and an abstract architecture for a DBNPM system. To demonstrate the feasibility of the approach, we also describe a realized prototype system based on the architecture and its application in a case study from industrial practice.

The structure of this paper is as follows. Considering the central theme of automating processes support within virtual enterprises, the state of the art in the area is discussed in Section 2 from both the ‘traditional’ workflow management point of view and from the currently ‘fashionable’ service-oriented point of view. In Section 3, we elaborate the concepts of DBNPM and IVE in more detail. Based on this elaboration, we identify the requirements for DBNPM in IVE and compare these to the state of the art. In Section 4, we take the identified requirements as the starting point for a structured design of the architecture for a DBNPM/IVE

support system. Section 5 treats realization of a software system that is based on the designed architecture. A prototype of this software system has been realized in the CrossWork IST project. In Section 6, we put the developed approach to the test by applying it to a case study from the automotive industry: we describe market context, realized prototype system and the dynamic network business process supported by the system. We conclude the paper in Section 7 by presenting the main observations from the work described and a brief outlook into future developments in the covered area.

2 THE STATE OF THE ART

In this section, we give an overview of the state of the art related to this paper. We first discuss the field of interorganizational (also called cross-organizational) workflow management. Then, we move our attention to the field of service-oriented computing. As a third main topic, we address the use of multi-agent systems for business process support.

2.1 Interorganizational workflow management

Workflow management technology has been around since the early nineties of the previous century and is receiving ample attention in both industry and research. Much of workflow management technology, however, focuses on intra-organizational workflows, assuming a possibly distributed, but homogeneous workflow management system in a trusted environment. When workflow management is extended across organizational boundaries, the complexity is heavily increased, however. Below, we present a few developments in interorganizational workflow management technology.

The WISE project (Workflow based Internet SERVICES) at ETH Zürich aims at providing a software platform for process based business-to-business electronic commerce [Alo99, Laz01]. In doing so, the project focuses on support for networks of small and medium enterprises. The software platform used in WISE is based on the OPERA kernel [Alo97]. WISE relies on a central workflow engine to control cross-organizational processes (called virtual business processes). A virtual business process in the WISE approach consists of a number of black-box services linked in a workflow process [Alo99]. A service is offered by an involved organization and can be a business process controlled by a workflow management system local to that organization – but this is completely orthogonal to the virtual business process. The lack of local process handling makes WISE unusable for our context. WISE does not envision automatic composition of global processes. In WISE, processes are manually designed using the Structware/IvyFrame tool [Lie98].

In the CrossFlow project, concepts and technology for workflow support in dynamic virtual enterprises have been developed [Gre00, Hof01]. In the context of this project, the formation of virtual enterprises is based on dynamic service outsourcing, as advocated in this paper as well. Service offerings and service requests are specified in electronic contract templates [Koe00], which are matched by a service matchmaker. An established electronic contract is the basis for the dynamic generation of a service enactment infrastructure [Hof00], based on workflow management technology. Although CrossFlow supports automated, dynamic setup of cross-organizational processes, the approach relies on an asymmetric service outsourcing paradigm, in which an organization outsources a predefined part of its business process to a service provider. This paradigm is too limited for the multi-party, peer-to-peer situation required in the CrossWork context. The generalization of the CrossFlow approach [Gre03] still relies on the service outsourcing paradigm. The three-level process framework from this approach is one of the basic ingredients of the CrossWork approach, though.

The embedding of workflow management mechanisms in enterprise resource planning (ERP) systems and supply chain management (SCM) systems is addressed in industrial products and research efforts (e.g. [Liu05]). To use this in an interorganizational setting, however, requires the integration of these ERP or SCM systems across the boundaries of organizations, which is far from trivial, certainly in the context of dynamic collaboration.

2.2 Service-oriented computing

Service-oriented computing (SOC) is currently a ‘hot’ topic. SOC promises flexible, dynamic, component-oriented interoperability between business functionality of autonomous organizations. The functionality of a service can be quite diverse, depending on the application domain, from very simple, e.g. the functionality to convert an amount of money from one currency to another, or very complex, e.g. the functionality to invoke complex business applications. SOC as a concept is usually closely linked to Web services as a technology. The Web service paradigm allows the dynamic composition of (business) application functionality using the Web as a medium [Alo04].

The Web services framework also offers possibilities for flexible process integration. The BPEL (Business Process Execution Language) process orchestration language [BPL06] provides possibilities for process integration; UDDI (Universal Description, Discovery, and Integration) [UDD05] offers possibilities for dynamism in collaboration. The Web services framework has two drawbacks that limit its direct usability in dynamic network process management. Firstly, it relies on black box process integration: if processes from third parties are used in a BPEL specification, their internal structures are opaque. The BPWS framework [Gr06a] proposes an approach to overcome this limitation. Ideas from this framework are used in CrossWork. Secondly, the Web service framework offers in BPEL a too low level of semantics for the specification of rich business processes. Therefore, we choose a richer language for dynamic network process composition. As will be explained in the sequel of this paper, we do use BPEL after composition has been completed as the basic language in the enactment subarchitecture of the CrossWork system – this to improve interoperability and portability.

2.3 Agent-based approaches

Only a few multi-agent systems (MASs) for business processes and workflows exist. Approaches like ADEPT and CONOISE [Jen00, Nor03] use software agents to implement business services. An overall business process is split into tasks which are then executed by agents. However, the overall business process needs to be given first. The system designer has to assign individual services (tasks) to agents and to verify that all necessary tasks are covered in order to implement the overall business process; no support for this is given by the software system. Other approaches [Huh01] use agents to coordinate business partners in a supply chain. The behaviour of each partner agent is specified with an interaction protocol. The overall business process is not explicitly defined, but can be seen as the sum of the interaction protocols.

A different approach combining multi-agent systems and workflows has been taken and implemented by Buhler and Vidal [Vid04, Buhl04]. In this approach, a BPEL workflow specification is given. Similar to the systems above, the individual tasks are implemented by software agents. In contrast to the approaches above, here a bridge between the BPEL standard and the FIPA standard is built (cf. [BPL06, FIP07]). In a follow-up system, Buhler and others [Buh05] have built an agent-based system the other way around. Here, FIPA discovery and inter-agent communication services are used to implement a Web service composition engine. Agents take over the job of planning and scheduling the execution of individual Web services. A distributed planning algorithm is implemented by the agents, which allows online discovery of agents and their services. The planning mechanism is based on a simple input/output match, and no user interaction on global level is possible. Currently, no workflow integration is done. Since BPEL also relies on the same Web service standards, the approach could be enhanced allowing agents to trigger different distributed and independent workflows in parallel. However, in this case no global workflow is given and the overall behavior of the system is of an emergent nature.

In contrast to these existing agent approaches, we use agents to construct the overall business process but use non-agent workflow enactment technology at execution time. This choice allows the use of existing industry-strenght enactment systems in our approach and ameliorates the integration with legacy systems.

3 DYNAMIC BUSINESS NETWORK PROCESSES IN INSTANT VES

In this section, we lay the foundation for the sequel of this paper by explaining and discussing its main concepts: business network process (BNP) and instant virtual enterprise (IVE). If business network processes are applied in IVE, we require dynamic business network process management (DBNPM). From the nature of these concepts, we next distill the requirements to automated systems supporting DBNPM in an IVE.

3.1 The BNP concept

In specific business domains (like logistics, insurance, car manufacturing, etc.), we see markets in which autonomous business organizations operate. Markets can be regional, but nowadays have an increasingly global character. In a market, a large number of business organizations can be present – depending on the domain and the geographical spread, this can vary from tens to thousands of autonomous organizations. Figure 1 shows a simple market, in which ellipses denote business organizations (the number of them is kept small for reasons of clarity).

Each business organization has its local business processes to reach its local (internal) business goals. Two (or more) local business processes in an organization may implement different local business goals (e.g., producing two different products) or may be different operationalizations of the same business goal (e.g., producing the same product with two different production procedures). Local business processes are illustrated in Figure 1 as simple connected process graphs within the boundaries of the business organizations – for reasons of clarity, most organizations are shown with one local process only (the two lowermost organizations each have two local processes).

Local business processes exist at conceptual, external and internal levels [Gre03]. The conceptual level defines the logical business view of a process, i.e., the structure of a process without constraints by local technological infrastructures or external (market) requirements. The internal level specializes a conceptual process definition such that it is adapted to the local technological infrastructure (e.g., the local enterprise information systems). The external level contains a projection of the conceptual-level specification of a process that is meant to be

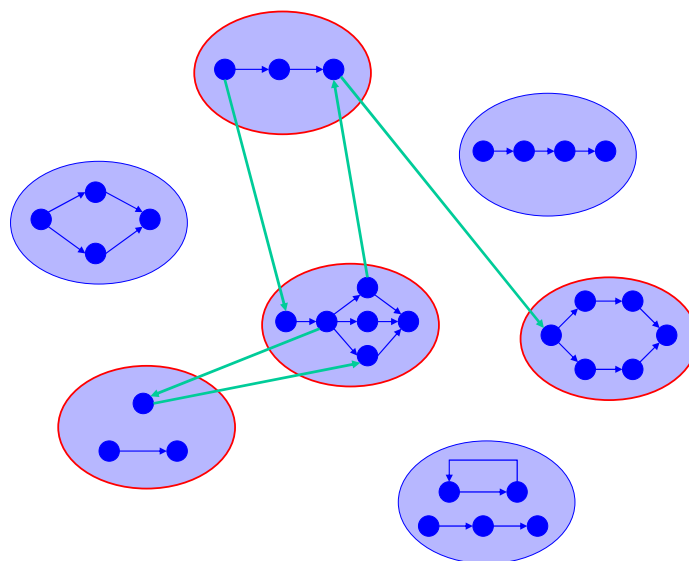


Figure 1: example business network process

externalized, i.e., made visible to other business organizations in a market. Hence, the external level is relevant for collaboration; more concretely, it allows synchronization of multiple local business processes of different business organizations.

Synchronization is required because – as discussed in the introduction of this paper – production of complex products or enactment of complex services implies business goals that cannot be realized by a single organization (and typically neither by just two or three or them). Therefore, networks of business organizations have to be formed within a market, consisting of nodes (members) that have capabilities that contribute to the overall business goal. Forming a network means very carefully selecting organizations such that the overall business goal can be reached, but no unnecessary capabilities (and hence organizations) are added.

The members in the network are arranged in a peer-to-peer topology. This means that the organizations collaborate at the same level, as opposed to a client/server topology in which organizations are organized hierarchically. The peer-to-peer collaboration requires fine-grained synchronization between participating autonomous parties. Note that one organization in a network may act as contact point to a client party (i.e., accepts orders) or that a network may have one member that synchronizes all other members (i.e., it functions as a ‘collaboration hub’), but that this does not imply hierarchy among the members of the network.

To operationalize the synchronization of the members in the business network, the local business processes (at the external level) of the members need to be connected into a global process, called a business network process (BNP). The BNP is created by adding the appropriate control flow connections between external level local processes of organizations participating in the BNP. This is illustrated in Figure 1 by the arcs between the organizations (crossing the boundaries of the ellipses).

As suggested by Figure 1, organizations are often not synchronized as black boxes showing no internal details, but based on the structure of their local, external-level processes – as required by the fine-grained interaction between organizations. Synchronization between two organizations is possibly bi-directional, meaning that the organizations may be waiting for each others events at different places in the global process. We can formulate the required interaction style in terms of the four interface classes defined in [Gre03]. A black box interaction style between members (as often used in Web service approaches) does not suffice in all situations. A glass box style, which permits other members to see the progress of process execution by a member, provides a basis for synchronization in a BNP. The half-open box style (as used e.g. in the CrossFlow project [Gre00]) allows finer interaction, but even this style is not always sufficient in this context as it only supports unidirectional control flow dependencies. An open box interaction style may be required, which allows for arbitrary control flow dependencies between local processes in a global process.

3.2 The IVE and DBNPM concepts

The BNP concept as introduced above can in principle be applied in a static context, in which predefined global business processes are collaboratively executed by stable virtual enterprises. Such stability can only function in static markets, in which goals to be reached by business networks do not change frequently. As we have discussed in the introduction, however, currently we live in a world with very dynamic markets. This implies frequently changing global business goals, which in turn require BNPs that are defined on-the-fly, based on the what and when of a specific place and time.

To support this dynamism, we introduce the concept of the instant virtual enterprise (IVE). An IVE is a virtual enterprise that is formed on-the-fly during business operation based on a selection of capabilities required for a specific business goal at a specific moment in time. Instant virtual enterprises have a peer-to-peer character as opposed to a client/server character. Application of the BNP concept in an IVE context leads to the concept of dynamic business network process (DBNP) and dynamic business network process management (DBNPM).

As suggested by its name, DBNPM is in two ways an extension of ‘classical’ business process management (BPM) or workflow management (WFM): it considers business processes that have an interorganizational network structure and it considers processes that are forged dynamically, i.e. on-the-fly during business operation. These extensions imply specific requirements to a system supporting the approach. These requirements are discussed in the following subsection.

3.3 System requirements for DBNPM in IVEs

Based on the description of dynamic business network process management (DBNPM) in instant virtual enterprises (IVEs) in the subsections above, we can describe the functionality required from a system that provides automated support for this approach. We formulate the functionality in terms of the following high-level functional requirements to a DBNPM/IVE system:

- RQ1. Given a global (IVE-level) business goal gg , the system can semi-automatically decompose gg into a structure of local (organization-level) business goals slg .
- RQ2. Given a structure of local business goals slg , the system can semi-automatically identify a set of organizations so in a business market such that the organizations in so together have the capabilities required to reach gg by implementing each of the local goals in slg .
- RQ3. Given a local business goal lg and an organization o , the system can semi-automatically obtain the specification of one or more external level local business processes of o that implement lg .
- RQ4. Given a set of local business processes slp , the system can semi-automatically compose the local processes in slp into an IVE-level business network process (BNP).
- RQ5. Given a BNP bnp , the system can validate process execution characteristics of bnp without actually enacting it in an IVE, where validation is interactively performed by a business process engineer.
- RQ6. Given a BNP bnp , the system can automatically map bnp to the distributed DBNPM system of an IVE.
- RQ7. Given a BNP bnp mapped onto the DBNPM system ds of an IVE, the system can automatically enact bnp on ds , where enactment includes providing end user interaction functions and process manager monitoring functions.
- RQ8. In the enactment of a given BNP bnp , the DBNPM system facilitates interaction with legacy (back-end) systems of the organizations enacting bnp .

Some of the above requirements include the qualification ‘semi-automatically’. The reason for this is the fact that fully automatic realization of these requirements is not directly feasible in most situations, a domain knowledge is required that is not available in machine-interpretable format. The system should, however, support migration of knowledge from human-oriented format towards system-oriented format, such that the level of automation can increase as domains are getting more formalized. Therefore, we include the following additional requirement:

- RQ9. For those system functions that rely on reasoning on the basis of domain-based knowledge, the system supports accumulation of this knowledge into knowledge stores that can be accessed by automated reasoning mechanisms.

Note that we have not included non-functional requirements in the above discussion. Non-functional aspects do play a role, however, when getting to the technical details of a system implementation. We address these issues in Section 5.

Clearly, the above list of requirements can be further refined to obtain a software requirements specification – this is, however, not in the scope of this paper. We choose to map the high-level requirements to a high-level system architecture, however. This is discussed in the next section. Functional details following from the requirements are treated later when discussing the functionality of the software modules in the architecture (see Section 5).

4 DESIGNING THE SYSTEM ARCHITECTURE

Taking the system requirements identified in the previous section as a basis, we present the design of a conceptual system architecture in this section. As the requirements imply a complex architecture, a clear design approach is essential to arrive at a well-structured architecture. The design approach applied in the sequel of this section consists of two main phases:

- a general clustering of functionalities is taken as a starting point,
- then, a stepwise refinement is performed.

The general clustering of functionalities is based on a very abstract separation of concerns. This separation of concerns is not specific for the architecture under design, but is a conceptual tool to obtain a starting point for an architecture design in which these clusters play major roles. We address this phase in Section 4.1.

The stepwise refinement is based on the result of the first phase and the requirements identified in Section 3.3. We show two aggregated refinement steps in this paper – a more detailed picture is presented in [CW05]. In the refinement steps, we use well-accepted design principles in the form of architecture patterns [Bus96]. The refinement steps are presented in Sections 4.2 and 4.3.

4.1 Starting with a separation of concerns

As discussed above, we start the architecture design with a high-level functional separation of concerns to provide a basis for an overall clustering of DBNPM functionality. We use an interrogative-based separation of concerns, where we apply the following four interrogatives:

- **What** is the operationalized goal of an IVE to be formed, given a global goal specification?
- **Who**, i.e. which organizations, can together constitute this IVE?
- **How**, i.e. by what business process, can this IVE indeed reach the operationalized goal?
- **With what** automated infrastructure can this business process be enacted?

The functional clusters related to the four interrogatives correspond to four consecutive phases in the formation and enactment of an IVE based on DBNPM. The fact that we have consecutive phases, leads to the observation that the functional clusters should be embedded into a *pipe and filter* architecture pattern [Bus96]. This leads to the architecture shown in Figure 2. In this figure, we see that the creation of a DBNPM-based IVE starts with a goal specification (specified by a client organization), which triggers a pipeline of four software modules corresponding with the four interrogatives.

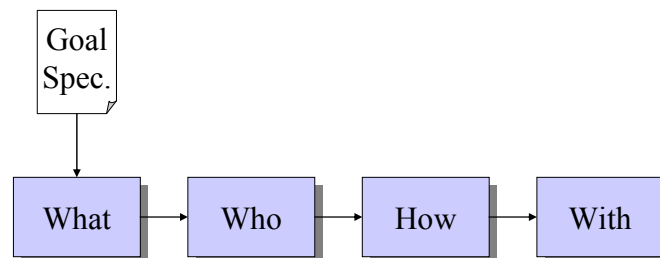


Figure 2: architecture after separation of concerns

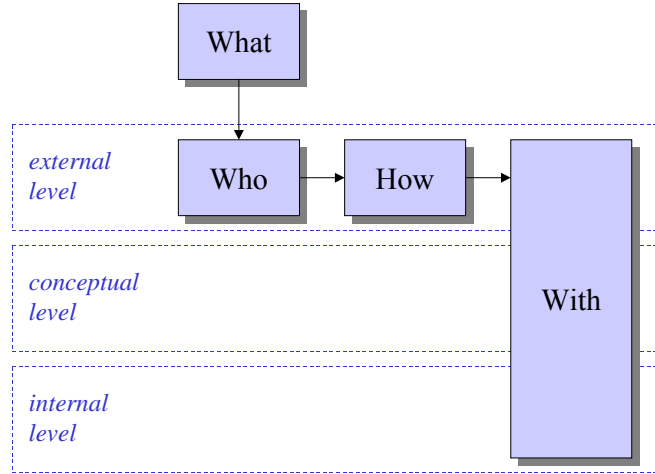


Figure 3: architecture after applying three-level framework

The architecture shown in Figure 2 does not identify any levels, i.e., it places all four modules at the same level. We obtain more structure, however, by also applying a separation of concerns with respect to abstraction levels. For this, we apply the three-level process framework for interorganizational, process-oriented collaboration [Gre03] that we used in the previous section to identify the BNP concepts. The result of the application of the framework is a layered architecture [Bus96] as shown in Figure 3.

In Figure 3, the layers are not yet fully crystallized: the ‘With’ module is spread across layers (which violates the *layers* architecture pattern). Hence, a further detailing of functionality is required to obtain a properly layered architecture. This detailing is described in the next subsection.

4.2 Refinement step one: detailing functionality

In this subsection, we further detail the architecture shown in Figure 3 by exploding the identified modules where necessary. This detailing is based on requirements identified in Section 3.3. To do this, we start with matching the requirements to the functional clusters in the architecture. The result of this matching is shown in Table 1.

	RQ1	RQ2	RQ3	RQ4	RQ5	RQ6	RQ7	RQ8	RQ9
What	X								X
Who		X							X
How			X	X	X				X
With						X	X	X	X

Table 1: matching architecture clusters and functional requirements

When analyzing the results in the table, we treat RQ9 separately, as this requirement refers to a very specific functional aspect (we revisit this in the next subsection). Looking at the other requirements, we see that the ‘What’ and ‘Who’ clusters each correspond to a single requirements, and hence do not need explosion at this level of abstraction. The ‘How’ and ‘With’ clusters each correspond to multiple requirements, so we explode these two clusters on this basis. Further functional details of the identified modules are explained in Section 5.

The functionality of the ‘How’ module concerns determining the business process in an IVE to achieve the overall business goal (RQ3-RQ5). This implies obtaining local business processes of members of an IVE and weaving them into a global business process, i.e., composing a global workflow from a number of local workflows. From an architectural point of view, the

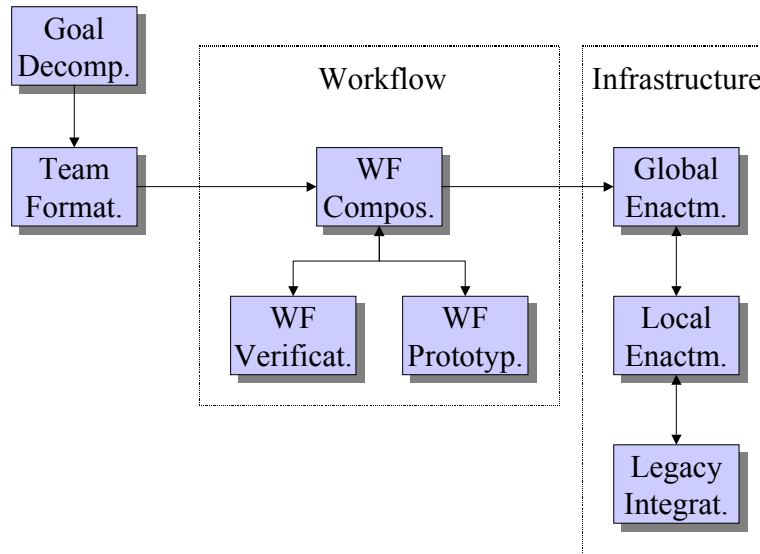


Figure 4: architecture after detailing functionality

functionalities related to RQ3 and RQ4 are interwoven, so we decide to allocate them into a single architectural module called Workflow Composition. Fulfilling RQ5 implies both static verification and dynamic prototyping of composed processes. From an architectural point of view, these are two separate functionalities, which we thus place in two separate functional modules: Workflow Verification and Workflow Prototyping. The result of the explosion is shown in Figure 4.

The functionality of the ‘With’ module concerns enactment of composed global business processes, taking into account the legacy situation at participating IVE members (RQ6-RQ8). The distinction between overall process management (interorganizational synchronization) at IVE level and local process management within scope of a single IVE member leads to the observation that we need to identify a Global Enactment module and a Local Enactment module in the detailed architecture. The coupling of the DBNPM system to back-end systems implies complex functionality that we allocate to a separate module: Legacy Integration. The general functionality of this module is to provide a common interface to execute operations in multiple, heterogeneous Enterprise Information Systems (EIS). The result of this explosion is also shown in Figure 4.

We can map the detailed architecture again to the three-level framework [Gre03], as we have done before for the functional clusters in Figure 3. Here, the modules resulting from the ‘With’ cluster require attention, as this cluster extends over multiple layers. As the Global Enactment module coordinates IVE members across their boundaries, we place this module at the external level. As the Local Enactment and Legacy Integration modules depend on the infrastructure existing at specific IVE members, we place these modules at the internal level. The result is shown in Figure 5

In our architecture, we do not include any enactment functionality at the conceptual level, as this would require double dynamic mapping between module states. The conceptual level is of interest, however: the design of local business processes within a specific IVE member takes place at the conceptual level and is next mapped to external and internal levels. For clarity, this is also shown in Figure 5 – but note that automated support for this design is not part of our approach (as the design of local business processes is not specific for DBNPM) and hence not discussed in this paper. We will see in the next subsection that local process specifications (the results of the design activity) are an essential element of the market knowledge required for the creation of IVEs.

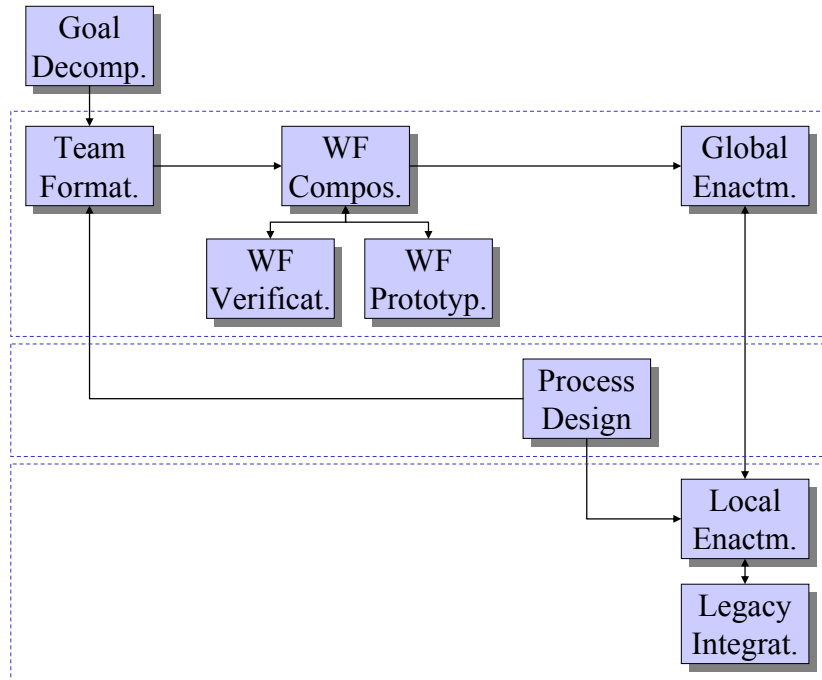


Figure 5: detailed architecture in three-level framework

4.3 Refinement step two: adding knowledge

As a DBNPM system has to perform high-level, complex tasks, it must be able to apply knowledge of IVE markets, IVE members, their local processes and information system infrastructure, etcetera. The ultimate goal of DBNPM development is to obtain a fully automated system, but the complexity of DBNPM prohibits reaching this goal in a short period of time. Therefore, a DBNPM system must be designed such that it can use knowledge supported by human users and knowledge stored in knowledge bases, such that a gradual transition from a mainly interactive to a fully automated system is possible – this has been listed in the requirements analysis as RQ9.

Equipping the architecture of Figure 4 with means to support this hybrid decision making means adding three kinds of functionality to it:

- dedicated, interactive user interfaces, through which users can feed decisions to the appropriate software modules.
- automated knowledge bases, which accumulate formalized knowledge in the context of a specific application domain (or market) about IVE formation and enactment.
- advanced automated reasoning logic in system modules identified in the architecture.

The advanced automated reasoning knowledge is not visible in the architecture at this aggregation level – we address this in the next section. We extend the architecture of Figure 4 first with the required user interfaces. As a business engineer is responsible for both goal decomposition and team formation, a single interactive user interface module (Formation User Interface) is used to interact with these two modules. A process engineer should be able to interact with the Workflow Composition modules (and via this with Workflow Verification and Prototyping); for this, we include the Workflow User Interface. Finally, an operations manager in an IVE must be able to monitor and control BNPs during their enactment. Therefore, we include a Monitoring User Interface. Note that the Local Enactment module also requires end

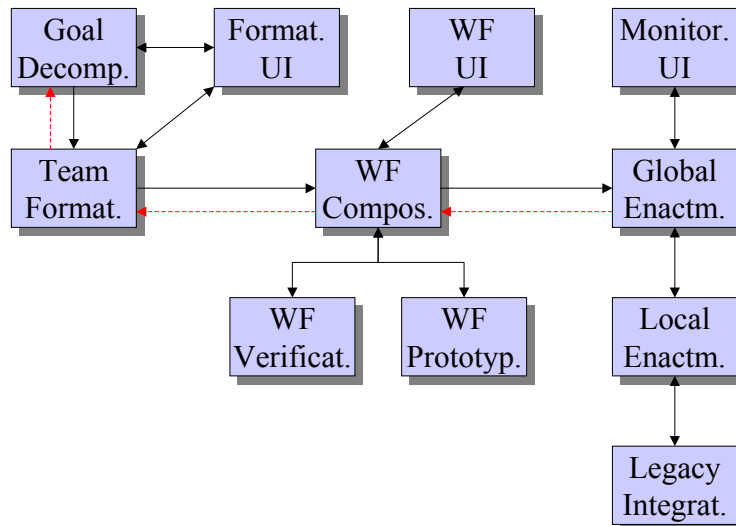


Figure 6: architecture with user interfaces added

user interaction; this is however not specific for DBNPM and is addressed later in this paper. The resulting architecture is shown in Figure 6.

Note that reasoning leads to decisions, and decisions may be negative. For example, the Workflow Composition module may decide that a correct BNP (global workflow) cannot be composed. In this case, the system must ‘backtrack’, i.e., go one or more steps back in the creation of an IVE. This is supported by interfaces between modules as indicated by the ‘reverse’ dotted arrows in Figure 6.

Next, we add knowledge bases to the architecture. We distinguish between the following knowledge bases:

- The product knowledge base contains general knowledge about the products in a specific application domain (as produced by IVEs) and their composition (comparable to a bill of materials); it is used by the Goal Decomposition module.
- The market knowledge base contains specific knowledge about organizations in a market (potential IVE members), their capabilities and their local processes; it is used by the Team Formation module to select potential members for an IVE.
- The infrastructure knowledge base contains knowledge about the back-end (legacy) information systems existing at specific organizations; this knowledge is used by the Team Formation module to avoid the composition of teams that are incompatible with respect to their local infrastructures.
- The workflow pattern knowledge base contains workflow specification patterns that are used in the composition of BNPs (global workflows) by the Workflow Composition module.

Figure 7 shows the architecture of Figure 6 extended with these four knowledge bases. This is the ‘final’ diagram in our architecture discussion. A more detailed description of the architecture and its design process can be found in [CW05].

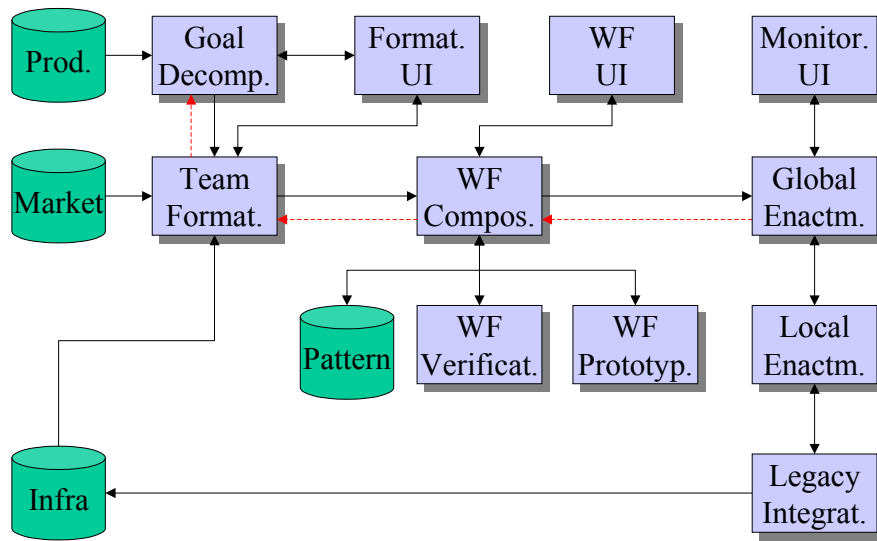


Figure 7: architecture with knowledge bases added

5 IMPLEMENTING THE SYSTEM COMPONENTS

In the previous section, we have described an abstract architecture of a (semi-)automated system for the support of DBNPM in IVEs. In this section, we devote our attention to the realization of a prototype system following this architecture. Before we map the components in the architecture to concrete software implementations, we first have to decide about the technological context, i.e., the software technology (platforms) to be used. After we have treated the choice of platforms, we discuss the realization of the components in the architecture, grouped by the technological context they use.

5.1 Choosing the software platforms

The choice of software platforms for the realization (embodiment) of the architecture is partly determined by non-functional requirements. These non-functional requirements can be seen as a complement to the functional requirements list in Section 3.3. Important requirements are:

- ease of realization of complex module functionality,
- support for complex interaction between architecture modules, and
- possibilities for future extension of a prototype system.

In choosing platforms, we are confronted with two ‘faces’ of the system. Firstly, the IVE ‘build time’ part of the system (goal decomposition, team formation, and workflow composition modules) requires a platform supporting high-level, knowledge-based reasoning. Secondly, the IVE ‘run time’ part of the system (global enactment, local enactment, and legacy integration modules) requires a platform supporting easy interoperability to existing process management technology and legacy systems. For this reason, we have chosen different platforms for the IVE build time and IVE run time subsystems, with an adapter in between.

For the IVE build time (setup) functionality, we have chosen multi-agent system (MAS) technology [Woo02]. MAS technology is well suited for the implementation of distributed decision making, reasoning and handling of knowledge (e.g. through the use of ontologies). Where non-agent technology is to be used (e.g. for workflow verification), we use agent wrappers to make this technology MAS-compliant. The JADE [JAD06] platform has been chosen as a concrete technology here.

For the IVE run time (enactment) functionality, we have chosen to use service-oriented technology. As coupling to existing systems is a main issue here, conformance to industry interoperability standards has priority. We use a Web service based infrastructure for the embedding of the enactment modules. Where necessary, we use Web service wrappers to encapsulate non-WS technology. For process specifications in the run time environment, we use standard BPEL [BPL06]. This choice also enables the use of a standard BPEL engine as a basis for global workflow enactment.

Given these platform choices, we describe the functionality of the architecture modules and their realization on the chosen platforms in the sequel of this section.

5.2 Goal decomposition and team formation

The aim of the goal decomposition and team formation stage is to form a team that is capable of achieving a stated goal. In a manufacturing context, this goal can be well-defined, for example an order specification may contain the full bill of material of a certain composite part, e.g. a dashboard in the automotive industry. In the general case, however, the goal would be quite vague, allowing open interpretations and substantial flexibility in the ways in which it could be

fulfilled. An example of the latter type of goal would be to create a seat for a new car model, allowing substantial flexibility and creativity within few constraints.

Algorithmic software solutions are unlikely to work well in such underdefined contexts. We have instead chosen to learn from the domain of design engineering (e.g., [Cha99]) and to base our software solution on flexible problem-solving approaches, which operate on the basis of a set of knowledge structures. These knowledge structures describe relevant aspects from the application and problem-solving domains, e.g., the domains of engineering design, cross-organizational process coordination and manufacturing processes. We use conceptualizations of these domains to inform our knowledge models. These models would then be enriched with information we know about the problem set as a goal, for example we may add constraints on the team members or the specific order instance containing a decomposition of a complex part into components plus all the processes necessary to assemble the complex part and produce its components. These knowledge structures are used as an input to the reasoning routines. They also inform what information is missing and hence should be collected from the user of the system before the goal can be decomposed.

The use of the knowledge-based problem-solving approach is complemented by the use of software agents [Woo02] as a core technology to organize the modules responsible for Goal Decomposition and Team Formation. Software agents represent the interests of different actors, and drive the problem-solving behavior of software. Their suitability is based on the match between core characteristics of systems based on software agents [Jen01] and specific features of the domain of interest:

- Agents' capability of autonomous goal-seeking behavior has a direct mapping onto the domain of team assembly, where autonomous partners are brought together in a team to pursue common goal in a concerted manner;
- The agents' ability of advanced communication and negotiation with other agents can facilitate the formation of innovative emergent teams, where the team composition depends on the outcome of negotiations and on slight variations of input conditions.
- Agents' reasoning abilities can be subjected to audit trails, and when agents implement systematic decision-making and evaluation techniques, important business-critical decisions can be audited and justified.
- The importance of domain knowledge in the chosen problem-solving approach fits well with agents' reasoning mechanisms which are also based on knowledge representations, often ontology-based.

The use of software agents and knowledge structures allows us to decompose the goal to sub-goals, whilst possibly seeking additional information from the user. This knowledge-guided decomposition would eventually result in specifications for parts and services for supplying and assembling these parts. For brevity we just use the term 'services' in the sequel.

Once we have identified the service specifications, we can attempt to find suppliers for these services from our databases of known suppliers. We may have a number of suppliers competing to provide a service, in which case we can apply systematic selection or negotiation approaches [Sha04, CN02]. We may be able to form our team using this centralized top-down approach; this would be quite a realistic scenario for saturated markets and well-specified goals allowing decomposition to detailed services.

In sparse markets, however, there may be no suppliers for the service we have identified. Also, the open-ended way in which our problem has been formulated may result in a set of services which are at a very high level of abstraction, such as 'design a seat'. In these cases, we use a novel bottom-up composition approach described in detail elsewhere [Car06]. In this approach, the service for which we seek suppliers is pinned on a notice board, where it can be observed by agents representing a community of service providers. These agents can identify whenever the service provider represented by them can provide a partial solution for a given

service, and, if they are interested to provide this service, they would display their partial solution on the notice board, thus inviting complementary partial solutions by other members of their community. Extending an existing partial solution is done with the agreement of the providers of that partial solution, thus any consortia thus formed should be able to work together.

Once a consortium can provide a full solution, the different alternatives are evaluated by the (human) ‘customer’ agent, which can select the ‘best’ solution according to some relevant criteria. The team thus selected is then passed onto the workflow composition module, described next.

5.3 Workflow composition, verification and prototyping

Composition algorithms for the construction of business network processes make use of data flow structures and patterns to determine control flows in the global workflow (i.e., to populate the open box interfaces). To allow a rich specification of local processes and composition of global processes, we use a dedicated process specification language, called eSML. eSML is a multi-aspect, XML-based process specification language that has been developed to support general sourcing [Nor06], of which DBNPM is a specialized variant.

The definition of eSML has been strongly influenced by the research on workflow patterns. For example, for specifying control flow, eSML uses operators that are based on existing control flow patterns found in commercial workflow management systems [Aa03a]. This way, the expressive power of eSML exceeds that of most commercial workflow languages by far. As a consequence, a specific composed global workflow can only be used in those workflow management systems that support all the patterns used in the specific global workflow. We have adopted BPEL as an enactment language to ensure interoperability and portability. Therefore, the constructed compositions only use patterns also present in BPEL [BPL06].

The actual composition process consists of two phases [Till05]. In the first phase, the global workflow is constructed by analyzing data flow dependencies between the given local workflows [Esh06]. In the second phase, the constructed global workflow is verified and validated. For this phase, advanced tool support exists [Ver04, Nor04]. The control flow part of eSML is based on XRL [Aa03b], which has a formal semantics in terms of Petri nets. This allows the application of state-of-the-art Petri net analysis techniques to analyze a global workflow. The correctness of a composed workflow, for example absence of deadlocks, can be automatically checked with an automated workflow verification tool. The global workflow can be prototyped using a light-weight workflow enactment system that uses XRL as its process specification language. We are currently extending this tool support by developing a tool that checks a global workflow for data flow constraints and general business rules and constraints. An example of such a business constraint is the requirement that the throughput time for a BNP is at most 15 days. The open character of the architecture allows the seamless integration of new verification and validation tools and services at any time

5.4 Workflow enactment

After a business network process has been composed and verified, it is ready for enactment by an instant virtual enterprise. Enactment of a BNP is based on a two-level mechanism, consisting of global process orchestration and local process execution. Both levels are designed such that they allow flexible enactment topologies through the use of remote workflow clients [CW05].

To assure interoperability with industry standard process enactment platforms, an industry-standard business process execution language (BPEL) [BPL06] is used for enactment. BPEL is an XML-based language, built on top of Service Oriented Architecture (SOA) and Web services specifications, which is used to define and manage long-lived service orchestrations or

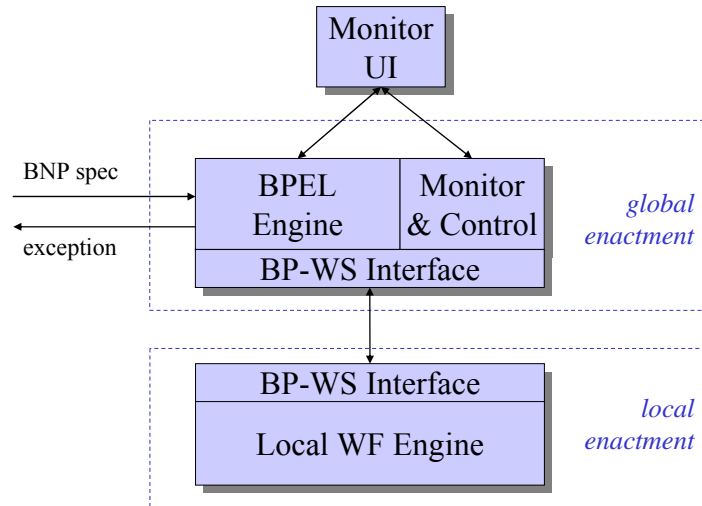


Figure 8: enactment architecture

processes. In BPEL, a business process is a large-grained Web service, which executes a control flow to complete a business goal. The steps in the control flow execute activities that are centered on invoking partner services (e.g. Web services specified in WSDL) to perform tasks and return results back to the process. The aggregate work, the collaboration of all the local services, is a service orchestration. The drivers for choosing BPEL in our approach are manifold. First, enterprises are evolving their SOA implementations from simple, fine-grained services, to more complex, large-grained services. Second, enterprises are employing service-oriented architecture strategies for integration. Third, in response to the first two items, vendors are creating integration and SOA infrastructure solutions that offer BPEL orchestration, and/or use BPEL for internal processing. Finally, the specification is maturing: BPEL 2.0, sponsored by OASIS, is expected to be available soon.

Figure 8 depicts the enactment architecture – it is an elaboration of the enactment subarchitecture as shown in Figure 7. The two levels of workflow enactment (global enactment and local enactment) are clearly identified in the enactment architecture. Next to this, we employ a monitoring user interface module.

The Global Workflow Specifications expressed in BPEL are obtained by automated translation of eSML specifications. For that reason, an eSML2BPEL translation submodule is included in the architecture. This submodule represents the bridge between the pre-enactment and the enactment phase. For the real-time enactment of the Global Workflow, a standard BPEL engine is used. It reads BPEL process definitions (and other inputs such as WSDL files) and creates representations of BPEL processes. When an incoming message triggers a start activity, the engine creates a new process instance and starts it. The engine takes care of persistence, queues, alarms, and many other execution details.

During the enactment of the global process and depending on the control flow, local partner Web services (representing local business processes) are invoked either asynchronously or synchronously to perform their job. A precondition is that local partners have already exposed their business processes as Web services and they have already exposed a WSDL interface. But this is not enough. In DBNPM, a web-service that represents a business process, in contrast with a ‘traditional’ Web service, is often required to offer external visibility to its internal process structure. For this purpose, we use the concept of a Business Process Web service (abbreviated to BP-WS), introduced by [Gr06a] which includes a business process specification and business process state that can be accessed externally. Access to specification and state are provided through a number of dedicated Web service interfaces (ports). [Gr06a] introduces four BP-WS classes following four control flow interface levels, which we use as a basis here. According to its internal business logic (or other initiatives) each local partner may decide to expose (or not)

its services by using the BP-WS classes. Local Partners specify selected internals of their business processes (a projection of the business process specification) to the outside world by using eSML (for workflow composition) and BPEL (for monitoring and control purposes).

The concept of developing business process specifications (in BPEL) for the local Web services enables the overall monitoring of the enactment at both local and global level. For that, statefull global BPEL specification files need to be combined with statefull local specification files. In our approach, we expand the capabilities of the global Enactment Engine by adding monitoring and control functionalities: a Monitoring and Control engine. The purpose of this engine is two-fold: to communicate with all specification and control ports of all local and global services in order to be aware of the combined status of execution and to distribute all control messages to the appropriate local services through the control ports.

5.5 Ontology and user interface support

The bottom-up approach of modeling the supplier network [We05a], as used in our approach, allows individual organizations and users to describe and use their personal styles of working. They can encode ‘how’ they would like to work in their ‘local’ workflow descriptions (see above). The ‘what’, i.e., the information objects needed for fulfilling a certain task, is stored in several ontological structures (as already mentioned above). This personalization of data models and workflow provides the flexibility that is required to conquer trends in the manufacturing industry like globalization and agility [Bro95, Ver02]. The decentralized nature of the overall system does require that organizations are enabled to design and maintain private data models at run-time.

The data structures are designed using a set of common core ontologies that are extended for the different implemented scenarios. Stalker et al. [Sta05, Sta06] gives details about the underlying ‘devolved ontology’ approach and its mathematical foundation. This approach allows a decentralized maintenance of different individual ontologies based upon a common core. Instantiations of these ontologies can automatically be translated across sub-domains. The two major ontologies used within our approach are the so called Market Ontology and the Product Ontology. The latter allows describing automotive products and projects, the former employees and suppliers.

Software implementation wise, the ontologies are encoded using JADE [Jad06] Ontology beans. These are java classes which allow employing java-reflection mechanisms by providing standardized naming of data access methods. Java reflection mechanisms are extensively used in the back- and front-end of our architecture. For storage of information at the back-end, the Ontology beans are disaggregated and the bean structure information and the content of the properties are stored separately. In order to visualize data on the front-end, ontology bean objects are recursively parsed using the java reflection mechanism and displayed in a generic manner. Using the reflection mechanism in this way allows extending and modifying the ontologies without the need to modify the back- or front-end but still have a working database and user interface.

Essential in the approach of gradual transition from ‘human’ knowledge to automated knowledge is a proper user interface set that connects the human and automated parts of the approach. To illustrate working with ontologies as described above, the screenshot in Figure 9 shows the user interface of a front-end with an instance of the product ontology (the screenshot is from the prototype system that we discuss in the next section). The pane on the left-hand side guides the users through the already above described steps. The center pane provides information about the problem at hand. In this case a product called ‘RUECKSCHLAGVENTIL’ is visualized. This instance is placed in a bill of material which can be browsed using the tree structure on the left hand side on the center pane. The right hand side of this pane provides further information about this concept instance. The bottom pane is

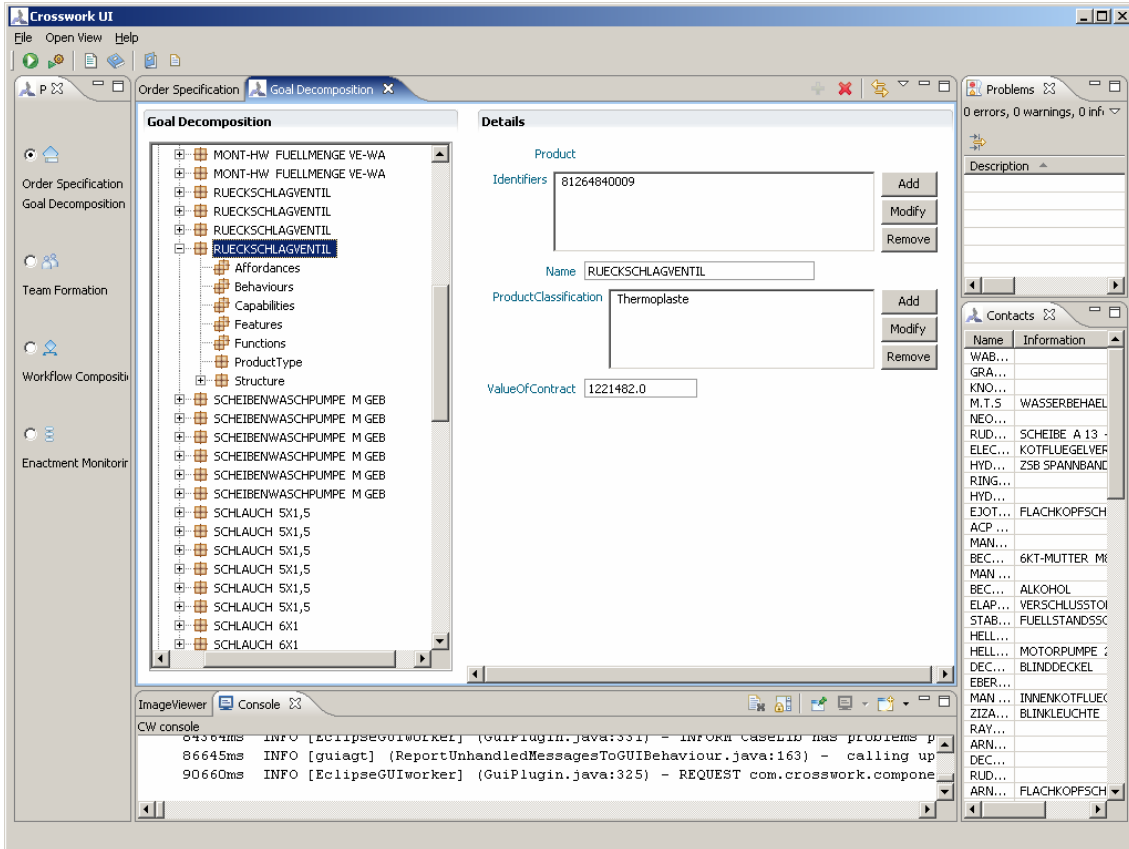


Figure 9: user interface visualizing ontology beans

used to provide additional user feedback and the right hand side provides access to information about suppliers.

Figure 10 shows another user interface, supporting the visualization of a workflow (allowing a process designer to interact with the workflow composition logic described in Section 5.4). In this case the global and the local workflows are shown. Here, in the top of the center pane the user is allowed to accept or reject a proposed global workflow. Rejecting it allows to select different team members, accepting it automatically deploys it on the workflow execution engine. The middle part of the center pane shows the overall workflow by visualizing the generated BPEL file. The bottom part gives detailed information on individual tasks. The right hand side allows browsing the structure of the workflow.

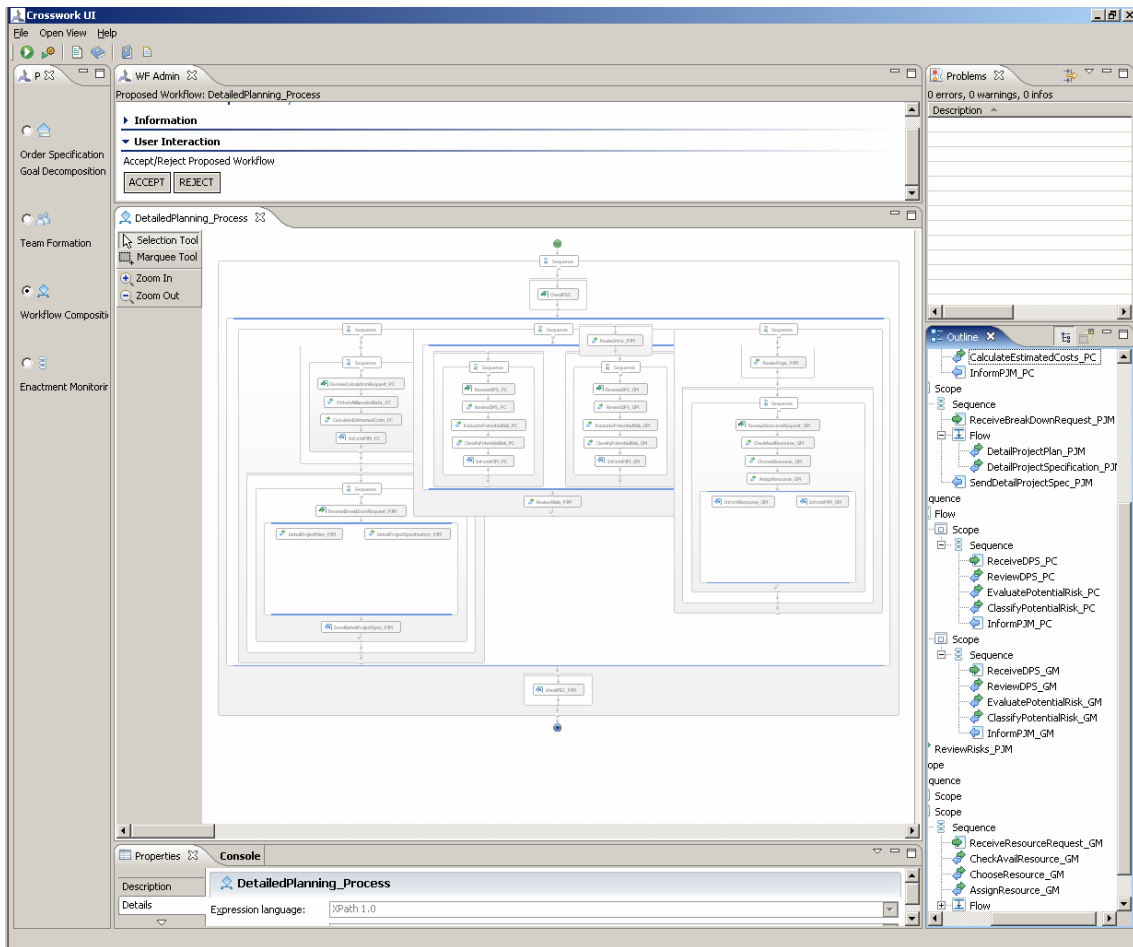


Figure 10: user interface visualizing a workflow

6 CROSSWORK: PUTTING THE APPROACH TO THE TEST

In this section, we discuss an application context for the DBNPM and IVE framework we have described in this paper so far. The application context is defined by the IST research project CrossWork [CW07], in which a number of industries from the software and automotive domains and a number of research partners form a consortium in which the use of DBNPM in IVEs is developed for the automotive industry. This section can thus be seen as a real-world case study towards the feasibility and usability of the DBNPM/IVE approach.

Below, we first describe the business-level developments in the automotive industry to clarify the business context in which the case study is embedded. After that, we discuss the case study from the CrossWork project. Although multiple case studies have been performed in this project, for reasons of clarity and brevity, we have taken a simplified version of one of them to illustrate the application of our approach in practice. For further and more elaborate case studies, the reader is referred to CrossWork project documentation.

6.1 Developments in the automotive industry

Business processes in the automotive sector are of a complex structure, typically spanning a number of organizations in a supply chain. The organizations in a supply chain are organized along an automotive supply pyramid. At the top of the pyramid, we find an OEM (original equipment manufacturer, i.e., brand car producer) that assembles cars. Below the OEM, we find a number of first tier suppliers that provide relatively large car parts (called 'systems') to the OEMs. One level down again, a larger number of second tier suppliers are positioned that supply modules (smaller car parts) to the first tier suppliers. At the bottom of the pyramid, we find the third tier suppliers that provide components to the second tier suppliers. Fourth tier suppliers provide raw materials. In the automotive sector, we typically find large numbers of SMEs that act as second and third tier suppliers, but also (to a limited extent) as first tier suppliers.

Within a supply chain pyramid, complex interorganizational business processes are enacted for design and production of car parts and complete cars. Such a process is often enacted by an OEM and a network of SMEs. Traditionally, the network and its processes are based on collaborations that are formed during lengthy meetings and negotiations between candidate members of the network. The overall design phase, which includes the design of the supply chain, often takes a period of three years. The interorganizational processes either consist of isolated local (intra-organizational) processes that heavily rely on vertical ad-hoc synchronization to cover the interorganizational aspects, or are predetermined and rigid.

In the past few years, we have seen a number of new developments in automotive sector that put the above situation under pressure [Bro95, Ver02, San01]:

- OEMs are pushing responsibilities down the supply pyramid to their first tier suppliers, thereby making collaborations in the pyramid more decentralized, and thus increasing the need for synchronization.
- Second tier suppliers are organizing into virtual enterprises to become virtual first tier suppliers and thus directly collaborate with OEM's, thus increasing the need for complex horizontal (intra-tier) synchronization.
- Increasing global competition forces automotive supply chains to become more agile and more efficient. Here, agility implies the means to set up new processes in networks much faster and cheaper than in traditional collaboration structures. It also means creating structures supporting efficient and flexible enactment of these processes.

The need for managing agility, complexity and efficiency requires automated support for dynamic business process management across automotive networks, thereby transforming a network of automotive suppliers into a Network of Automotive Excellence (NoAE).

The CrossWork project aims at designing concepts, architecture and technology supporting (semi-)automated business process management in NoAEs. CrossWork is a European research project in the 6th IST framework that started its work in early 2004 and is completed in early 2007. It unites a number of important players in the automotive industry, the software industry and in academic research [CW07].

6.2 The CrossWork case study

In the case study, an OEM requests the production of a water tank from one member of a cluster of automotive suppliers. This member, which acts as the main contractor (also referred to as ‘systems integrator’) towards the OEM, typically does not have the capability to produce the complete water tank itself. Consequently, it has to create an instant virtual enterprise (IVE) by first finding additional suppliers in the cluster that can assist in fulfilling the OEM's request, and next define a business network process (BNP) that coordinates all the supplier processes and interacts with the OEM. The CrossWork system helps to achieve this by providing semi-automated support for setting up the IVE and the BNP in its context.

In the first step, the goal ‘produce water tank’ is decomposed into subgoals. The knowledge structure for decomposing this goal is stored in the product knowledge base. In the knowledge structure, there are two general aspects. The product aspect focuses on the decomposition of the product into subcomponents, and thus is similar to a Bill of Material. The service aspect concerns the services needed to deliver the request. The output of this step is a structured set of components and services. In this description of the case study, we do not focus on the service aspect. For example the shipping of components is considered to be a responsibility of each partner – hence, logistics processes are not monitored by the IVE in this case study. Figure 11 shows the product decomposition, shown as a ‘bill of material’ (BOM) consisting of automotive parts and related services. A part can have several variants. The BOM lists all required parts for all variants. An IVE must be able to produce all variants of a product.

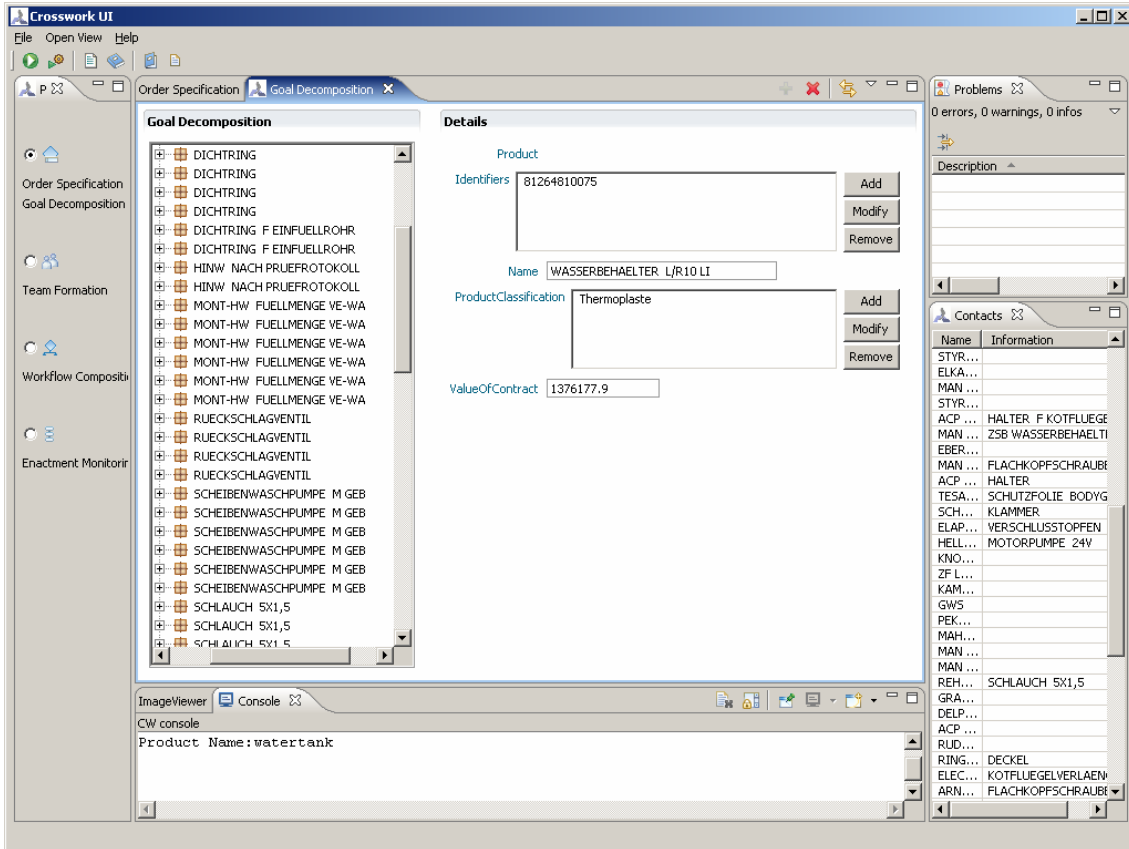


Figure 11: user interface of goal decomposition module

In this case, the design-for-assembly paradigm is used, since all components are already existing and specified in the knowledge structure. However, CrossWork also supports product development. In that case, an end user has to guide the goal decomposition by interacting with the CrossWork system. The resulting knowledge structure can be stored in the product knowledge base.

In the next step, a team has to be assembled. Based on the market knowledge base, the team formation module retrieves all partners that can produce or deliver one or more of the components that have been identified in the first step. Next, a team is assembled from this set of potential supplier partners using different team formation strategies (see Figure 12). One team out of the list of possible teams is expanded in this figure. Each team member is capable to fulfill at least one of the tasks (named “Producing_XXX”) shown. The team members in this example are selected because of their good performance profile, which is recorded in a so called scorecard. The scorecard details the performance of the supplier according to different aspects, like logistics and production quality. In this case, the main contractor is assumed to have a central role, so the constructed team consists of suppliers for the main contractor. The main contractor will perform the assembly of the parts delivered by the suppliers to produce complete water tanks.

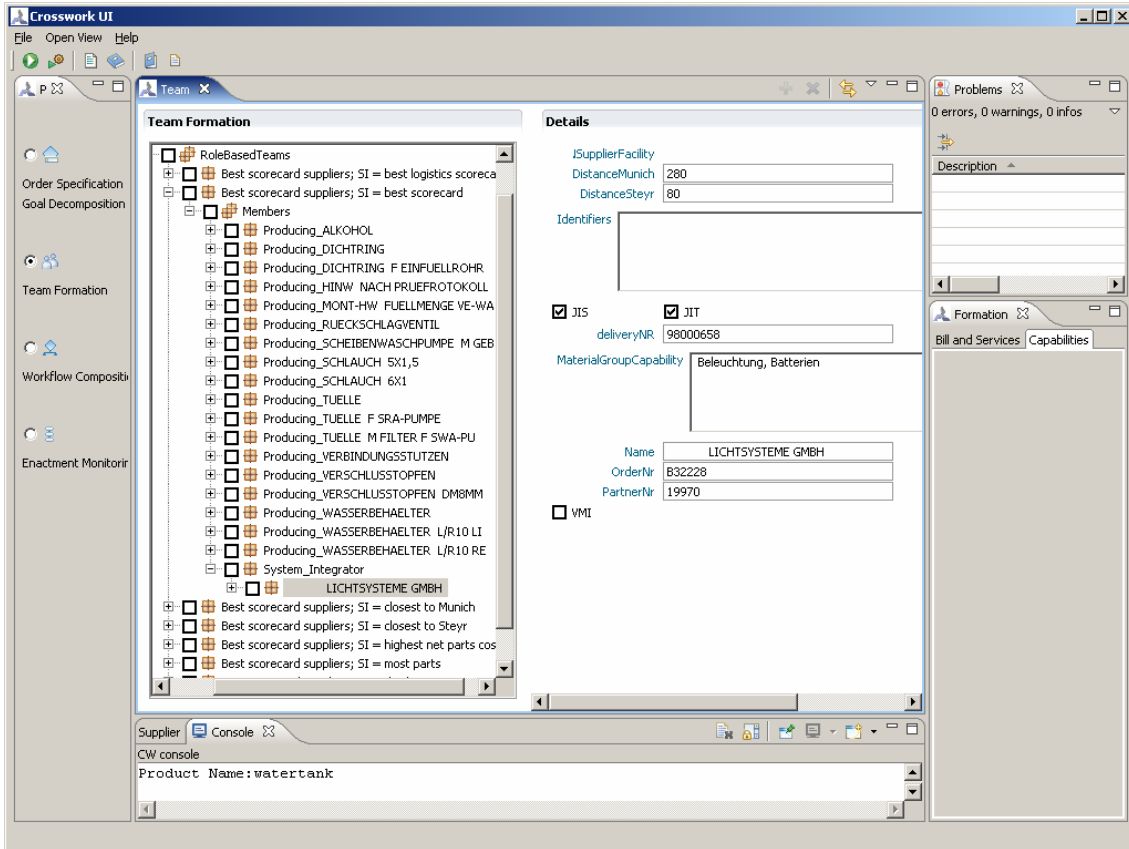


Figure 12: user interface of team formation module

In the third step, the global business process of the IVE is defined by composing the business processes of the individual partners. Each of the parts of the water tank is produced by a particular supplier using a specific local process. A local process may need certain input, to be delivered by other local processes, and may produce certain output, to be delivered to other local processes. By analyzing these data flow dependencies between activities, a BNP is formed [Esh06]. The formed BNP is then verified (e.g. checked for soundness) and eventually translated into the enactment language.

The output of this phase is a BPEL process, shown in Figure 13. Here, we see a BNP having five parallel branches, in which components for a water tank are produced by suppliers to the main contractor, followed by the assembly of the water tank by the main contractor itself. The assign-tasks in the BNP are needed to transfer data from one activity to the next. Note that in this case, the local processes are simple one-activity processes. This leads to rather trivial workflow composition and a BNP with a simple overall control flow. In practice, the local processes can be much more complex, therefore requiring more complex control flows in the BNP – and therefore putting more emphasis on automated support for workflow composition.

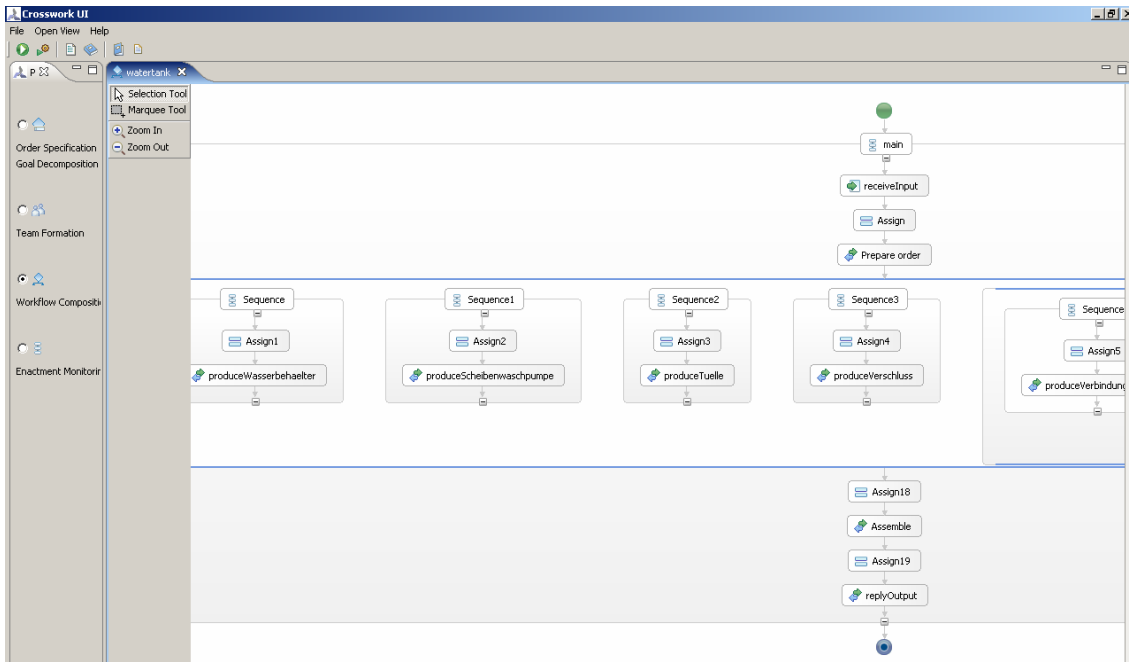


Figure 13: user interface of workflow composition module

In the fourth step, the composed BNP is enacted by the CrossWork enactment infrastructure to actually set the IVE to work, i.e., produce water tanks. The global enactment engine, located at one of the members of the IVE, coordinates local workflow engines located at one or more members (as we explain in the sequel of this section, members not owning a workflow engine can be linked in using remote client technology).

6.3 Implementation of the prototype system

In Section 5, we have described the software components in the architecture for supporting DBNPM in an IVE in generic terms, i.e., mostly independent of specific technology choices. In this subsection, we make things more concrete by describing the specific implementation choices made for the realization of the CrossWork prototype system. This description gives the reader an idea of the spectrum of technologies required for the realization of a DBNPM/IVE system.

One of the central technologies needed is a language in which we can specify collaboration primitives and their composition. For this, we have designed the electronic Sourcing Markup Language (eSML) [Nor07]. eSML is an XML-based language allowing the specification of collaboration scenarios from multiple perspectives. To describe local and global processes during the build phase of an IVE, the eXtended Routing Language (XRL) [Nor04] is used as a process specification sublanguage embedded in eSML.

To match current industry-standard execution platforms, we use BPEL [BPL06] as the global process execution language. Consequently, we employ an eSML2BPEL translator to translate eSML specifications to BPEL specifications to bridge the build time and enactment time subsystems. The translator has been realized using XSLT technology [XSL07]. ActiveBPEL [ABP07] is used as the global process engine. The ActiveBPEL engine is an open source implementation of a BPEL engine, written in Java. The ActiveBPEL engine can be used in any standard servlet container such as Tomcat [Tom07].

The i.Perform workflow management system [Exo07] is used as the local process engine. To limit the complexity, the CrossWork prototype of the enactment subsystem focuses on a selection of the interaction classes between global and local workflows: the black box, the glass

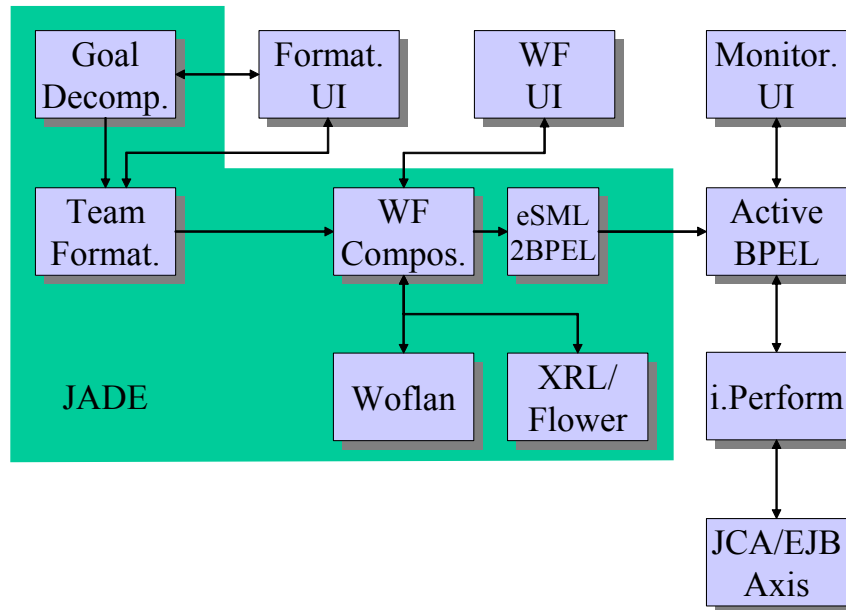


Figure 14: software modules in prototype system

box, and the half-open box interaction classes [Gr06a] – this means that completely free interaction patterns (open box class) are not supported. We use Woflan [Ver04] as the workflow validation tool and the web-based XRL/Flower system [Nor04] as the workflow prototyping system.

The build-time subsystem of the CrossWork prototype system is built on an agent-oriented platform. We have chosen JADE [JAD06] as the MAS platform on which the goal decomposition, team formation and workflow composition functionality is implemented. As discussed before, knowledge structures are implemented as Java Beans for use with JADE.

The implementation of the Legacy Integration module relies on a number of technologies. The Java-based J2EE Connector Architecture (JCA) technology solution [Sun07a] is used for connecting application servers and enterprise information systems (EIS) as part of enterprise application integration (EAI) solutions. It complements Web services and BPEL in a service-oriented architecture (SOA) environment [Erl05]. The Legacy Integration module uses JCA resource adaptors to connect to several EIS. Enterprise Java Beans (EJB) [Su07b] are used as a server-side component that encapsulates the business logic of an application. The Legacy Integration module uses Apache Axis for two purposes: to provide a Web service interface to the module and to develop client classes to use Web service connectors, mainly used to connect to .Net platforms.

An overview of the main technology choices is given in Figure 14, which is a concretization of the conceptual architecture in Figure 6 (reverse flows have been omitted here for the sake of clarity).

One of the issues in an industrial IVE environment (like the automotive domain) is the fact that there are many small and medium enterprises (SMEs) around that do not all own workflow management technology to enact their local processes. Therefore, we have chosen a local workflow engine (in the case of the prototype i.Perform [Exo07]) that offers a remote workflow client enactment architecture. In this way, IVE members that do not have a local workflow engine can use the local engine of another partner in the IVE. This latter partner hence operates as a workflow application service provider (ASP) to the former partner.

Figure 15 shows a simple example remote client topology. Here we see that IVE members A, B, and C do have local workflow engines (WFE), but member D does not. The global workflow is enacted on the engine of member A, which is connected to clients at A (local) and B (remote). Local workflows are enacted at the engines at B and C, with clients at A and B respectively C and D.

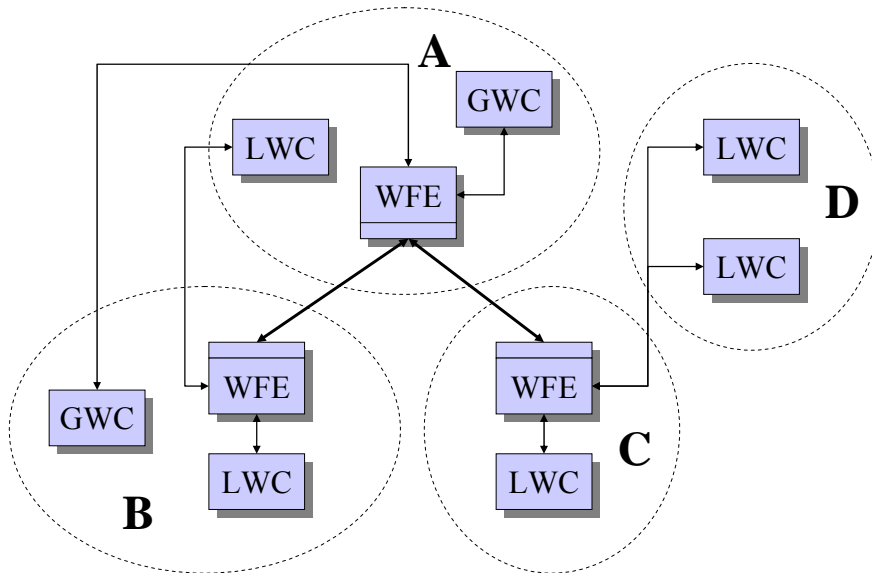


Figure 15: remote client enactment architecture

7 CONCLUSIONS AND OUTLOOK

In this paper, we have introduced the concepts of business network process (BNP) and instant virtual enterprise (IVE) and combined these into dynamic business network process management (DBNPM). From the description of the concepts at the business level, we have formulated high-level requirements as the basis for an automated system supporting DBNPM. For such a system, we have described a high-level architecture and discussed the functionality of the modules in this architecture. To demonstrate the feasibility and usability of the described approach, we have presented a case study including a real-world application scenario from the automotive industry and a prototype system implementation conforming to the architecture.

With the DBNPM approach, we extend the current state of the art in B2B process management. The approach has two elements that in combination make it stand out with respect to other approaches. Firstly, the DBNPM approach focuses on dynamic, multi-party market scenarios, in which complex instant virtual enterprises are created and dismantled to follow market movements. Secondly, DBNPM as presented in this paper provides a true end-to-end approach: it covers the entire spectrum from high-level, global business goals down to low-level, local business processes.

Adopting a DBNPM approach opens up new ways of doing business in complex, dynamic markets. Existing cooperative networks can be made more efficient and agile, thereby offering new levels of adaptivity to changing market conditions. New forms of cooperative networks can emerge that would not have been feasible without the structure and automated support offered by DBNPM, either because the complexity of the networks is too great for manual handling, or because the lifecycle of the networks is too short to allow unnecessary human involvement in setup and operation of the network.

The ideas presented in this paper have largely been developed within the context of the European CrossWork project. To be applicable in a broad spectrum of domains, the details of the approach need to be extended in future work. As explained in this paper, DBNPM allows a gradual transition from mainly manual decision making to fully automated decision making. To allow this transition, a big task lies with formalizing domain knowledge and transferring it into knowledge bases that DBNPM modules can effectively use.

Acknowledgements. The authors acknowledge the work of all members of the CrossWork project, whose work forms the basis for this paper. Sven Till is thanked especially for his help with producing the screen shots. Sven Till and Jochem Vonk are thanked for proof-reading this paper.

8 REFERENCES

- [Aa03a] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, A.P. Barros; *Workflow Patterns*; Distributed and Parallel Databases, Vol. 14, Nr. 3, 2004; pp. 5-51.
- [Aa03b] W.M.P. van der Aalst, A. Kumar; *XML Based Schema Definition for Support of Inter-organizational Workflow*; Information Systems Research, Vol. 14, Nr. 1, 2003; pp. 23-46.
- [ABP07] *Active BPEL Web Site*; www.activebpel.org; accessed 2007.
- [Alo97] G. Alonso, C. Hagen, H.J. Schek, M. Tresch; *Distributed Processing over Stand-alone Systems and Applications*; Procs. 23rd Int. Conf. on Very Large Databases; Athens, Greece, 1997; pp. 575-579.
- [Alo99] G. Alonso, U. Fiedler, C. Hagen, A. Lazcano, H. Schuldt, N. Weiler; *WISE: Business to Business E-Commerce*; Procs. 9th Int. Workshop on Research Issues on Data Engineering; Sydney, Australia, 1999; pp. 132-139.
- [Alo04] G. Alonso, F. Casati, H. Kuno, V. Machiraju; *Web Services – Concepts, Architectures and Applications*; Springer, 2004.
- [BPL06] OASIS Web Services Business Process Execution Language (WSBPEL) TC; *Web Services Business Process Execution Language Version 2.0*; available at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel; accessed 2006.
- [Bro95] J. Browne, P.J. Sackett, J.C. Wortmann; *Future Manufacturing Systems - Towards the Extended Enterprise*; Computers in Industry, Vol. 25, 1995, pp. 235- 254.
- [Buh04] P. Buhler and J. M. Vidal; *Enacting BPEL4WS Specified Workflows with Multiagent Systems*; Procs. Workshop on Web Services and Agent-Based Engineering; New York City, USA, 2004.
- [Buh05] P. Buhler, D. Greenwood, A. Reitbauer; *A Multiagent Web Service Composition Engine*; Procs. First International Workshop on Engineering Service Compositions; Amsterdam, The Netherlands; IBM Research Report RC23821; IBM Research Division, 2005.
- [Bus96] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal; *Pattern-Oriented Software Architecture (Vol. 1)*; Wiley, 1996.
- [Car06] M. Carpenter, N. Mehandjiev, I.D. Stalker; *Flexible Behaviours for Emergent Process Interoperability*; Process Integration of Collaborative Enterprises (PINCET), 15th IEEE WETICE'06, Manchester, UK, 2006.
- [Cha99] C. B Chapman, M. Pinfold; *Design Engineering – A Need to Rethink the Solution Using Knowledge Based Engineering*; Knowledge-Based Systems, Volume 12, Issues 5-6, 1999, Pages 257-267.
- [CN02] *FIPA ContractNet Interaction Protocol Specification, Dec 2002*; available from <http://www.fipa.org/specs/fipa00029/>, last accessed Dec 2006.
- [Cor02] F. von Corswant, P. Fredriksson; *Sourcing Trends in the Car Industry: A Survey of Car Manufacturers' and Suppliers' Strategies and Relations*; International Journal of Operations & Production Management; Vol. 22, No. 7; 2002; pp. 741-758.
- [CW05] *Global Architecture*; CrossWork Deliverable D4.1; 2005.

- [CW07] *CrossWork Web Site*; www.crosswork.info; accessed 2007.
- [Erl05] T. Erl; *Service-Oriented Architecture (SOA): Concepts, Technology, and Design*; Prentice-Hall, 2005.
- [Esh06] R. Eshuis, P. Grefen, S. Till; *Structured Service Composition*; Procs. Internatuional Conference on Business Process Management 2006, Lecture Notes; Springer; 2006; pp. 97-112.
- [Exo07] Exodus Web Site; www.exodus.gr; accessed 2007.
- [FIP07] FIPA – Foundation for Intelligent Physical Agents; Standards are available at <http://www.fipa.org>; accessed 2007.
- [Gre00] P. Grefen, K. Aberer, Y. Hoffner, H. Ludwig; *CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises*; Int. Journ. of Computer Systems Science & Engineering, Vol. 15, No. 5, 2000; pp. 277-290.
- [Gre03] P. Grefen, H. Ludwig, S. Angelov; *A Three-Level Framework for Process and Data Management of Complex E-Services*; Int. Journal of Cooperative Information Systems; Vol. 12, No. 4; World Scientific; 2003; pp. 487-531.
- [Gr06a] P. Grefen, H. Ludwig, A. Dan, S. Angelov; *An Analysis of Web Services Support for Dynamic Business Process Outsourcing*; Information and Software Technology; Vol. 48, No. 11; Elsevier; 2006; pp. 1115-1134.
- [Gr06b] P. Grefen; *Towards Dynamic Interorganizational Business Process Management (keynote speech)*; Proceedings 15th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises; Manchester, UK; 2006; pp. 13-18.
- [Hof00] Y. Hoffner, H. Ludwig, C. Gülcü, P. Grefen; *Architecture for Cross-Organisational Business Processes*; Procs. 2nd Int. Worksh. on Advanced Issues of E-Commerce and Web-Based Information Sys-tems; Milpitas, CA, USA, 2000; pp. 2-11.
- [Hof01] Y. Hoffner, S. Field, P. Grefen, H. Ludwig; *Contract Driven Creation and Operation of Virtual Enterprises*; Computer Networks, Vol. 37, No. 2, 2001; pp. 111-136.
- [Huh01] M. Huhns, L. Stephens; *Automating Supply Chains*; IEEE Internet Computing; Vol. 5, No. 4; 2001; pp. 90-93.
- [JAD06] JADE Board; *Java Agent DEvelopment Framework*; Available at <http://jade.tilab.com/>.
- [Jen00] N. R. Jennings, P. Faratin, T. J. Norman, P. O'Brien, B. Odgers, J. L. Alty; *Implementing a Business Process Management System using ADEPT: A Real-World Case Study*; International Journal of Applied Artificial Intelligence, Vol. 14, No. 5, 2000; pp 421–463.
- [Jen01] N. R. Jennings and M Wooldridge; *Agent-Oriented Software Engineering*; Handbook of Agent Technology (ed. J. Bradshaw) AAAI/MIT Press. 2001
- [Koe00] M. Koetsier, P. Grefen, J. Vonk; *Contracts for Cross-Organizational Workflow Management*; Procs. 1st Int. Conf. on Electronic Commerce and Web Technologies; London, UK, 2000; pp. 110-121.
- [Laz01] A. Lazcano, H. Schuldt, G. Alonso, H. Schek; *WISE: Process Based E-Commerce*; IEEE Data Engineering Bulletin; Vol. 24, No. 1, 2001; pp. 46-51.

- [Lie98] H. Lienhard; *IvyBeans - Bridge to VSH and the project WISE*; Proc. Conf. of the Swiss Priority Programme Information and Communication Structures, Zürich, Switzerland, 1998.
- [Liu05] J. Liu, S. Zhang, J. Hu; *A Case Study of an Inter-Enterprise Workflow-Supported Supply Chain Management System*; Information & Management, Vol. 42, Nr. 3, 2005; pp. 441-454.
- [Max04] G. P. Maxton, J. Wormald; *Time for a Model Change: Re-engineering the Global Automotive Industry*; Cambridge University Press, 2004.
- [Nor03] T.J. Norman, A. Preece, S. Chalmers, N. R. Jennings, M. Luck, V. D. Dang, T. D. Nguyen, V. Deora, J. Shao, W. A. Gray, and N. J. Fiddian; *Conoise: Agent-based Formation of Virtual Organisations*; Proc. 23rd SGA International Conference on Innovative Techniques and Applications of Artificial Intelligence; 2003; pp. 353-366.
- [Nor04] A. Norta; *Web Supported Enactment of Petri-Net Based Workflows with XRL/Flower*; Proc. International Conference on Application and Theory of Petri Nets 2004; pp. 494-503.
- [Nor06] A. Norta, P. Grefen; *A Framework for Specifying Sourcing Collaborations*; Proceedings European Conference on Information Systems 2006; Göteborg, Sweden, 2006.
- [Nor07] A. Norta; Ph.D. Thesis; Eindhoven University of Technology, 2007.
- [Op05a] S. Oppl; C. Stary; *Towards Human-Centered Design of Diagrammatic Representation Schemes*; In Proceedings of the 4th International Workshop on Task Models and Diagrams for User Interface Design, Gdansk, Poland, 2005
- [Op05b] S. Oppl and G. Weichhart; *Requirements for Collaborative Process Design*; In Auinger A. (eds): Workshop-Proceedings der 5. fachübergreifenden Konferenz Mensch und Computer 2005, Linz, Sept. 4th-7th, 2005.
- [OWL06] I. Herman and J. Hendler; *Web Ontology Language OWL / W3C Semantic Web Activity, revision 1.28 of 2006/10/27*; available on <http://www.w3.org/2004/OWL/>, accessed 2006.
- [San01] L. M. Sanchez, and R. Nagi; *A Review of Agile Manufacturing Systems*; International Journal of Production Research; Vol. 39, 2001; pp 351-360.
- [Sha04] S.E.Shaikh, N.Mehandjiev; *Multi-Attribute Negotiation in E-Business Process Composition*; WETICE'04 , pp.141-146, 2004.
- [Sta05] I.D. Stalker, N.D. Mehandjiev, M.R.J.Carpenter, and A. Gledson;. *Dynamic Knowledge Management in Open Multiagent Systems*; Proceedings of AMKM 2005, Workshop of AAMAS 2005. July, 2005.
- [Sta06] I.D. Stalker, N.D. Mehandiev; *A Devolved Ontology Model for the Pragmatic Web*; Proc. 1st International Pragmatic Web Conference; Stuttgart, Germany, September, 2006.
- [Sun07a] *J2EE Connector Architecture*; Sun Developer Network; <http://java.sun.com/j2ee/connector/>; Sun Microsystems; accessed 2007.
- [Sun07b] *Java Platform, Enterprise Edition (Java EE)*; Sun Developer Network; <http://java.sun.com/products/ejb/>; Sun Microsystems; accessed 2007.

- [Till05] S. Till, R. Eshuis, P. Grefen; *A Workflow Formation Architecture for the Automotive Sector*; Proceedings 2nd InterOp Workshop (in EDOC'05); Enschede, The Netherlands, 2005.
- [Tom07] *Apache Tomcat*; Apache Software Foundation; <http://tomcat.apache.org/>, accessed 2007.
- [UDD05] OASIS UDDI Specification TC; *UDDI Version 3 Specification*; 2005; available at <http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm>; accessed 2005.
- [Ver02] F.B. Vernadat; *Enterprise Modeling And Integration (EMI): Current Status And Research Perspectives*; Annual Reviews in Control, Vol. 26, 2002, pp. 15-25.
- [Ver04] H.M.W. Verbeek, W.M.P. van der Aalst, A. Kumar; *XRL/Woflan: Verification and Extensibility of an XML/Petri-net-based Language for Inter-organizational Workflows*; Information Technology and Management Journal, Vol. 5, Nr. 1-2, 2004; pp. 65-110.
- [Vid04] J. M. Vidal, P. Buhler, C. Stahl; *Multiagent Systems with Workflows*; IEEE Internet Computing; Vol. 8, Nr. 1, 2004; pp. 76–82.
- [We05a] G. Weichhart, K. Fessl; *Organisational Network Models And The Implications For Decision Support Systems*; In: Horáček, P.; Šimandl, M.; Zitek, P. (Eds.): Preprints of 16th IFAC World Congress. Praha, 2005
- [We05b] G. Weichhart, S. Oppl, and T. Waefler; *Flexible and responsive cross-organisational interoperability*. In Taisch M. and Toben K.D. (eds.): Advanced Manufacturing - An Ict & Systems Perspective; November 2005.
- [Woo02] M. Wooldridge; *Introduction To Multi-Agent Systems*; John Wiley & Sons, 2002.
- [XSL07] *XSL Transformations (XSLT) Version 1.0*; W3C; <http://www.w3.org/TR/xslt>;, accessed 2007.