Decentralized business process modeling and enactment: ICT architecture topologies and decision methods

Bernhard Bauer[1], Jörg P. Müller[2], and Stephan Roser[1]

[1] Programming Distributed Systems Lab
University of Augsburg
Universitätsstrasse 14, D-86135 Augsburg, Germany
{bauer|roser}@ds-lab.org
[2] Department of Informatics
Clausthal University of Technology
Julius-Albert-Str. 4, D-38678 Clausthal-Zellerfeld, Germany
joerg.mueller@tu-clausthal.de

Abstract. Multiagent systems have been proposed in the literature as a suitable architectural and implementation approach for cross-enterprise collaboration, due to their support for decentral decision-making and peer-to-peer coordination, loosely coupled interaction, modeling support for the notion of electronic institutions, and built-in adaptability mechanisms. While we agree with this general view, we argue that different application domain and different market constellations require different types of architecture. In this paper we look at the specific problem of selecting an information and communication technology (ICT) architecture for cross-enterprise business process (CBP) design and enactment. Therefore we identify three important architectural patterns for CBP enactment. We then propose a decision method for architecture selection based on the analytic hierarchy process (AHP) approach. Finally we illustrate the method by applying it to two application scenarios with differing characteristics. Robustness of the decision method is analyzed by performing a sensitivity analysis.

1 Introduction

Under the pressure of globalization, companies are urged to constantly adapt to new market situations and and competitors innovations. Focusing on their core business and core competencies, they engage in cross-enterprise business processes (CBPs) with new partners all over the world in ever changing constellations. Companies are organized into global networks and outsource those activities that can be performed quicker, more effectively, or at lower cost, by others.

These developments create new challenges for enterprise information and communication technology (ICT), requiring ICT systems to support constantly changing enterprise collaboration relationships and to create application systems

that support or automate business process enactment starting from business level descriptions and models of CBPs.

Multiagent systems [18] have been proposed in the literature as a suitable architectural and implementation approach for cross-enterprise collaboration [19, 20], due to their support for decentral decision-making and peer-to-peer coordination, loosely coupled coordination, modeling support for the notion of electronic institutions [14], and built-in adaptability. However, while we agree that the distributed and sometimes decentral topologies require some kind of multiagent organization principles, we argue that different application domains and different market constellations require different types of system architecture.

Let us look at the following example of collaborative product development in the automotive sector (see [35]). The scenario (see Figure 1) describes the interaction between an automotive Original Equipment Manufacturer (OEM) and its supplier network consisting of multiple tiers of suppliers, during the process of Strategic Sourcing. Strategic Sourcing is an early step within Cooperative Product Development, where OEMs set up strategic partnerships with the larger (so-called first-tier) suppliers with the aim of producing specific subsystems (e.g., powertrain, safety electronics) of a planned car series. In the use case considered for this paper, the OEM shares Requests for Quotations with its first-tier suppliers (1). First-tier suppliers serve as gateways to the supplier network; specifications are reviewed and conditions negotiated with second-tier suppliers (2), and feasibility of the requests are checked. First-tier suppliers then issue quotes or suggest changes to the OEM (3). This cycle is repeated until all parties agree on a feasible specification. Finally, first-tier suppliers submit quotes to the OEM.

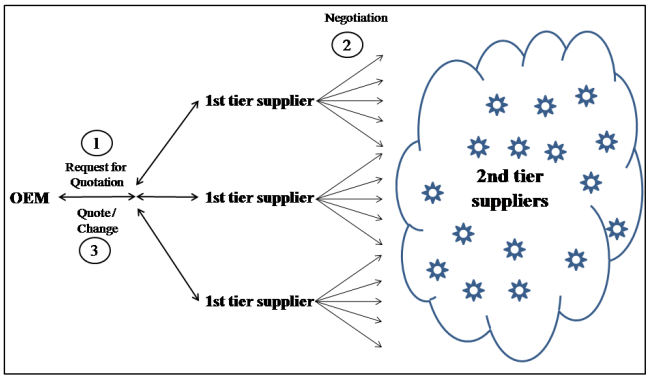

Fig. 1. Application example: Collaborative product development

Figure 1 gives a high-level (business-level) model of the cross-enterprise business processes involved in this scenario. Trying to map this model into an executable ICT model, several non-trivial questions need to be answered regarding

the ICT architecture. In particular, the question that we address in this paper is the following: What is the most appropriate architectural choice to model the interactions between the OEM and the 1st tier suppliers? A decentral, peer-to-peer messaging architecture where each role in each process instance is mapped into an agent-like entity to run and control it? An architecture with a central broker (e.g. located at the OEM) that centrally enacts and controls the cross-enterprise business processes? Or a mixture of both, a decentral broker architecture where each enterprise provides a publicly visible instance (agent) to control and coordinate their business process roles while hiding other, private, elements.

Thus, we can see using this scenario that there are different architectural choices / paradigms possible for underlying information and communication technology (ICT) system design. Intuitively, none of these choices is per se better than any other; making the right decision depends on a number of environmental characteristics (called contingencies in [10]). In this paper we propose a model for decision support suitable for enterprise architects to derive an appropriate supporting architecture paradigm for a given use case / application domain. Assuming some model-driven design paradigm [17, 4], we start from business level models and (semi-automatically) transforms these models into (platform-independent) ICT models (so-called PIMs); these PIM models are then subject to further transformation via platform-specific models (PSMs) down to the code-level.

In this paper, we address the problem of selecting a suitable ICT architecture at the PIM level, thus resulting in an architecture-driven approach to CBP modeling and enactment. The contribution of this paper is fourfold: First, we identify three important architectural patterns for CBP enactment; second we propose a decision method for architecture selection based on the analytic hierarchy process (AHP [31]); third, we show the applicability of the method by applying it to application scenarios with differing characteristics; fourth, we investigate the robustness of the decision method by performing a sensitivity analysis.

The paper is structured as follows: In Section 2, we provide the background on service-oriented architecture, model-driven engineering, and decision methods. Section 3 presents three ICT-level architecture patterns for CBPs. The decision method is proposed in Section 4; Section 5 analyzes the applicability of our method to two application scenarios. The paper ends with a discussion and outlook in Section 6.

2 Background

2.1 Model-driven engineering

Software engineering currently witnesses a paradigm shift from object-oriented implementation towards model-driven implementation. This carries important consequences on the way information systems are built and maintained [4]. Model-driven Engineering (MDE) treats models as first class artifacts, which are used for modeling and code generation. This raises the level of abstraction

at which developers create and evolve software [16] and reduces complexity of software artifacts by separating concerns and aspects of a system [17]. Thus MDE shifts the focus of software development away from the technology domain towards the problem domain. Largely automated model transformations refine (semi-)automatically abstract models to more concrete models or simply describe mappings between models of the same level of abstraction. In particular, transformation engines and generators are used to generate code and other target domain artifacts with input from both modeling experts and domain experts [32]. MDE is an approach to bridge the semantic gap that exists between domain-specific concepts encountered in modern software applications and standard programming technologies used to implement them [6].

Two prominent representatives of MDE are the OMG's Model Driven Architecture (MDA) and the software factory initiative from Microsoft.

In MDE, models and model transformations, which can be also treated as models, embody critical solutions and insights to enterprise challenges and hence are seen as assets for an organization [21]. Assets are artifacts that provide solutions to problems, should be reusable in and customizable to various contexts.

2.2 Service-oriented multiagent architectures

In recent years, a new generation of integration solutions has been developed under the service-oriented paradigm, which lends itself to develop highly adaptable solutions and to reuse existing applications. In a service-oriented world, sets of services are assembled and reused to quickly adapt to new business needs. However, service-orientation does not provide an integration solution by itself. Service-oriented integration introduces the concept of service (which can be implemented through Web Services) to establish a platform-independent model with various integration architectures. Service-oriented architecture (SOA) can be realized by an agent-oriented architecture. However, agents have additional features, services usually not have, like high-level, speech-act based communication, pre-defined interaction protocols (e.g. the ContractNet protocol or auction mechanisms) and goal-oriented composition of agents. In other words, agents are more sophisticated services. See also [33].

Service While, from an economic point of view, a service is the non-material equivalent of a good sold to a costumer, we use the term service from an ICT point of view, where a service is seen as a business or technical functionality. We define service *"as a well-defined, self-contained function that does not depend on the context or state of other services"* [5]. Service-orientation is based on this concept of service.

Agents Software agents are computer systems capable of flexible autonomous action in a dynamic, unpredictable and open environment [18]. These characteristics give agent technology a high potential to support process-centered modeling and operation of businesses. Consequently, starting with ADEPT [19], there have been various research efforts of using agent technology in business process

management. However, the focus of ADEPT was on communication and collaboration in business process management. It was not geared to being a directly usable business process support platform. Migrating agent technology successfully to business applications requires end-to-end solutions that integrate with standards, that preserve companies' investment in hardware, software platforms, tools, and people, and that enable the incremental introduction of new technologies. The OMG has set up a new standardization effort called *UML Metamodel and Profile for Service* (UPMS). The UPMS RFP requests a services metamodel and profile for extending UML with capabilities applicable to modeling services using a SOA[3]. Within this standardization effort a new RFP is prepared for the integration of agent technology in a service-oriented world. This can provide a first step towards solving this problem.

2.3 Cross-enterprise business processes

Many people and organizations participate in the construction of a software system, and impose different concerns and requirements on the system, in particular in CBPs. Business considerations determine non-functional qualities that must be accommodated in the system architecture. Quality attributes like availability, modifiability, performance, security, testability, usability, or business qualities are orthogonal to functional attributes describing the system's capabilities, services, and behavior. Since quality attributes are critical to the success of a system, they must be considered throughout design, implementation and deployment. Beyond these quality attributes, costs e.g. for hardware, software licences, and software development have to be considered when choosing the right architecture. In our work we investigate how service-oriented or agent-oriented architecture of software systems for CBPs can be derived from business level descriptions. The architecture variants and model transformations we describe are independent of functional attributes, since they can be applied to (nearly) any models describing CBPs. We investigate how three architecture variants for realizing CBPs in service-oriented software systems can be derived from business level descriptions and how to evaluate the right architecture for different contexts, thus supporting the enactment of high-level CBP specifications.

Orchestration & Choreography Orchestration and choreography describe two complimentary notions of a process. In orchestration a central entity coordinates the execution of services involved in a higher-level business process. Only the coordinator of the orchestration is aware of this composition. Choreography describes the interactions of collaborating entities (e.g. services or agents), each of which may have their own internal orchestration processes. These interactions are often structured into interaction protocols to represent the conversation between the parties. [27] An important distinction between orchestration and choreography is the fact that orchestration is generally owned and operated by a single organization while in a choreography no organization necessarily controls the collaboration logic [11].

[3] see http://adtf.omg.org/adptf_info.htm

Process modeling In process modeling it is common to distinguish between an internal and an external view of business processes. Depending on the viewpoint, a process is described either as an executable, abstract, or collaborative process: The internal view models the 'how' of a business process from the modeler's view. As the flow of an *executable process* [26] is described from the viewpoint of a single process coordinating its sub-processes, this is often referred to as process orchestration. *Abstract processes* model the external view on and the 'what' of a business process. Each process specifies the public interactions it performs in relation to its roles in collaborations. A *collaborative process* describes the collaboration between abstract processes in the case of process choreography. The collaborations between the involved parties are modeled as interaction patterns between their roles from the viewpoint of an external observer.

2.4 ICT architecture variants for CBP enactment

Service-oriented integration solutions can be categorized by their topology (see Figure 2). In a purely decentralized MAS topology services of the participating organizations implicitly establish the collaborative process through direct message exchange; this is a realization of choreography. In a hierarchical topology a controller service defines the steps necessary to achieve the overall goal and maps these steps to services provided by the contributing organizations; this is kind realization of orchestration. However, in many cases, a mixture of hierarchical and decentralized MAS topology, i.e. a heterogenous topology, is used to realize complex multipartner collaborations [23].

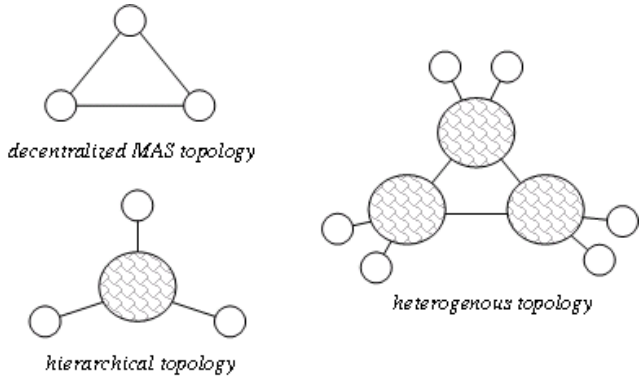

Fig. 2. Coordination topologies

2.5 Architecture evaluation and decision methods

Architecture evaluation Scenario-based ICT architecture evaluation is used to determine quality of software architecture. In architecture evaluation methods like ATAM, SAAM, or ARID [1, 8] quality attributes are characterized by scenario descriptions.

Quality attributes are part of the non-functional requirements and therefore properties of a system. They can be broadly grouped into two categories [9]. Qualities like performance, security, availability, and usability are observable via execution at *run-time*, and qualities like extensibility, modifiability, portability, or reusability, which are not observable via execution [3].

According to Bass et al. [1], scenario descriptions consist of a *stimulus* (a condition that needs to be considered when it arrives at a system), a *source of stimulus* (some entity that generates the stimulus) , an *environment* (the stimulus occurs within certain conditions), an *artifact* (the part of the system that is stimulated), a *response* (the response is the activity undertaken after arrival of the stimulus) and a *response measure* (defines how the result of the response is measured). General scenarios [2] are applicable to many software systems and have architectural implications; they establish sets of scenarios which are configured to the respective application domain (for which evaluation is performed) by varying the expected response value scales of the scenarios.

To be able to decide how good a quality attribute or a scenario is supported by a software architecture pattern and to compare architecture patterns, it is crucial to understand how an architecture influences quality attributes. According to Bass et al. [1] architects use so-called tactics to achieve quality attributes. A tactic is a design decision that influences the control of a quality attribute. The software architecture patterns described in this article make use of the following tactics (non-exclusive list; for detailed description see also [1] p.99ff): *Maintain semantic coherence*, *anticipate expected changes*, *generalize module*, *restrict communication paths*, *use an intermediary*, *maintain existing interfaces*, and *hide information*.

Tactics are used by an architect to create a design using design patterns, architectural patterns, or architectural strategies. An architect usually chooses a pattern or a collection of patterns designed to realize one or more tactics. However, each pattern implements multiple tactics, whether desired or not. The following list provides an overview of architecture patterns, design patterns, and design principles used to realize the above described tactics (non-exclusive list compiled from [1], [12], [11], and [13]): *Wrapper*, *broker*, *abstraction*, *loose coupling*, and *orchestration*.

Analytic Hierarchy Process The Analytic Hierarchy Process (AHP) [31] is a decision making approach, which decomposes a decision problem into a hierarchical network of factors and subfactors. Factor decomposition establishes a hierarchy of first level and second level factors cascading from the decision objective or goal. AHP applies pairwise comparisons to the factors and the alternatives in the decision making process. Pairwise comparisons lend themselves to solving problems with limited number of choices, where each choice has a

number of attributes and it is difficult to formalize some of those attributes. Finally the ratings of the second level factors are aggregated to first level factors and the final rating.

Contingency theory The contingency theory for organizations [10] is used to rationalize how the various aspects of organizations' environment (called *contingency factors*) influence organization structure. It suggests, that there is no unique or best way to organize an organization, but the design of an organization and its systems must 'fit' with its environment. The *"organizational effectiveness results from the fitting characteristics of the organization, such as its structure, to contingencies that reflect the situation of the organization"* [10, p.1]. *"Contingency theory (...) sees maximum performance as resulting from adopting, not the maximum, but rather the appropriate level of the structural variable that fits the contingency. Therefore, the optimal structural level is seldom the maximum, and which level is optimal is dependent upon the level of the contingency variable"* [10, p.4]. Translating this into the terms of companies and their business systems, a maximum of centralization, decentralization, or some of the ICT system architectural qualities like modifiability, security, etc., will seldom yield maximum performance of an ICT system for the overall business goals.

3 Architecture paradigms for CBPs

In Sections 2.4 and 2.2 we have introduced the abstract topologies for CBP enactment and described how service-orientation and agents fit together. Now we have a closer look at these coordination architectures and how they can be applied to realize service-oriented integration solutions. These architectures are used to control the conversation flow between the participating organizations. In an agent world this is comparable to interaction protocols. For the description of the coordination architecture we assume, that each organization willing to participate in a cross-organizational collaboration supported by ICT systems, has a set of *elementary services (ES)*, which are as far as possible realized by agents. These elementary services are application, business, or hybrid services. In our descriptions we also assume without loss of generality, that the elementary services are realized as process services, so that we can use the distinction between executable and abstract process. Nevertheless, elementary services could be realized by arbitrary code fragments. An elementary service can only be a controller service with regard to the organizations' internal service composition, but not with regard to the collaboration process. *Cross-organizational business processes (CBPs)* represent the conversation flow and message exchange between the organizations participating in the collaboration (in particular in an agent communication language).

– *Brokerless architecture*: A brokerless coordination architecture (see Figure 3) can be used to realize the decentralized MAS topology, where messages are exchanged directly between the elementary services of the participants

as usual in an agents world. Due to the mutual exchange of messages the elementary services depend on each other. Control flow logic of CBPs is realized by the executable process of the participants' elementary services. Changing the business protocol would result in changing multiple elementary services, i.e. their executable processes. Further, the abstract process of the elementary services are directly exposed to the collaboration space and therefore are directly accessible by entities outside enterprise boundaries.

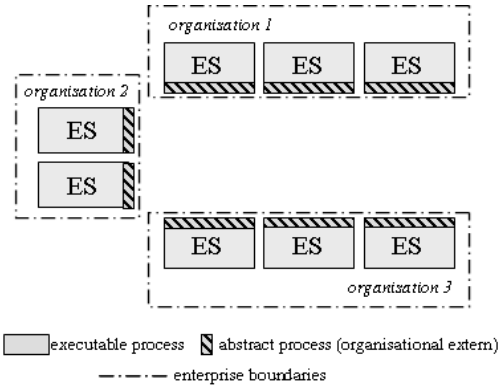

Fig. 3. Brokerless architecture

- *Central broker architecture*: Figure 4 depicts the central broker coordination architecture. Messages are no longer exchanged directly between the elementary services, but over a central broker component, which is realized by a controller service. The controller service is a process that orchestrates the elementary services of the participating organizations. It acts as a global observer process coordinating the partners as well as making decisions on the basis of data used in the CBP. In the case of a change to the CBP protocol's messages and semantics, only the broker process needs to be modified. Since the broker process is not necessarily owned by one of the participating partners, organizations may hide their elementary services from their collaborators. However, they have to reveal them to a third party instead.
- *Decentral broker architecture*: The *decentralized broker architecture* introduces elements of the decentralized MAS topology in the hierarchical topology of the central broker architecture. It splits the single broker component into several controller processes jointly providing the broker functionality (note the boundaries in Figure 5). Each organization provides one controller service, also called *view process (VP)*, which orchestrates the organization's internal elementary services. Messages across organizational boundaries are only exchanged by the view processes, which encapsulate the elementary services. In this architecture the elementary service can be seen as kind of *private processes (PP)*.

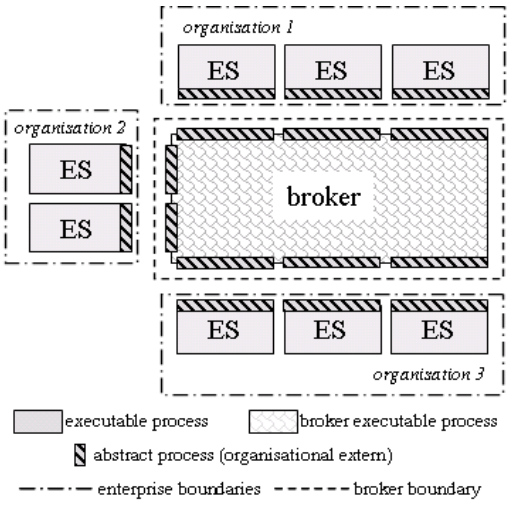

Fig. 4. Central broker architecture

4 A method for evaluation of ICT architecture applicability

This section presents an evaluation and decision method that helps to select appropriate ICT architectures for CBPs enactment. The evaluation method takes into account the trade-offs between coordination structures, which are implemented by the ICT system architectures in terms of coordination costs and vulnerability costs (see [24]). As visualized in Figure 6 the evaluation model distinguishes between quantitative factors, that are measurable by concrete figures (objective factors), and qualitative (subjective factors), which are difficult or impossible to measure. Coordination costs to establish and maintain communication links between collaborating patterns are included as quantitative factors in terms of software, hardware, and labor in the evaluation model. Coordination costs like costs for exchanging messages between collaborating partners are taken into account by qualitative factors. Vulnerability costs, which are "the unavoidable costs of a changed situation that are incurred before the organization can adapt to a new situation" [24], are qualitative factors in the evaluation model.

To be able to compare the architectural CBP approaches in the face of architectural decisions, it is necessary to get a quantitative measure from qualitative actors. Thus, we apply, extend and customize the multi-criteria decision model of Ghandforoush et al. [15], which is a modified version of Brown and Gibson's model [7]. As it is a quantitative model, it is useful for selecting one alternative from a given set of alternatives based on quantitative and qualitative factors. Figure 6 depicts the design of our multi-criteria decision model developed to evaluate ICT architectures for CBP enactment. The rating of the quantitative factors is determined by the means of cash-flow analysis of the predicted costs.

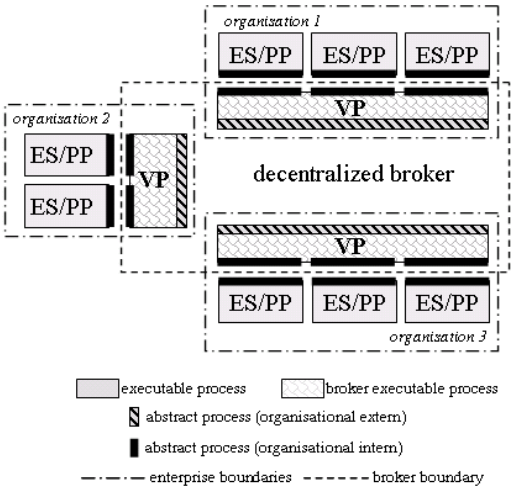

Fig. 5. Decentral broker architecture

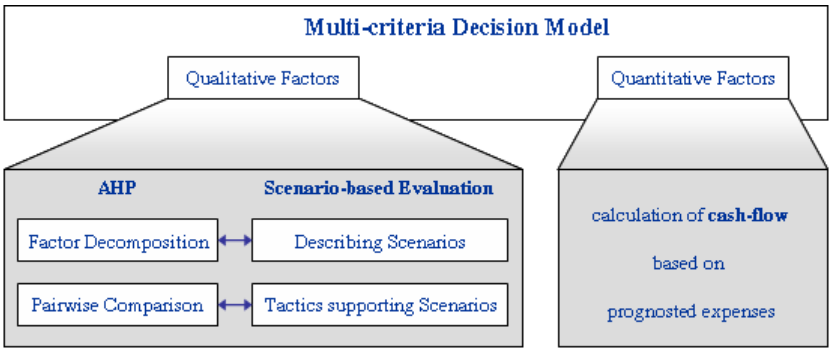

Fig. 6. Multi-criteria decision model for ICT architectures

For rating the qualitative factors we combine AHP and scenario-based software architecture evaluation methods. First, based on the AHP, the factors, which have to be considered in the evaluation, are determined by decomposing the evaluation problem and arranged in a hierarchy decomposition tree. The factors are described by the means of quality attributes and scenarios. Rating the scenarios and the alternatives is done by pairwise comparison. The ratings, i.e. the pairwise comparisons, are based upon how good the alternatives realize tactics supporting the respective scenarios.

4.1 Multi-criteria evaluation and decision model

The multi-criteria evaluation and decision distinguishes between objective (quantitative) factors and subjective (qualitative) factors.

- *Objective factors* are evaluated in monetary terms, and as such are easily quantifiable. Our quantification is based on the cash flow approach and therefore on the discounted present value. The evaluation model considers costs for software, hardware and labor.[4]
- *Subjective factors* are characterized by the fact that they are qualitative measures that typically cannot be quantified. When evaluating software architecture, quality attributes and scenarios are measures in qualitative terms.

The underlying principle of the model is to combine the two evaluation factors into a common evaluation measure. This requires that quantitative considerations and qualitative considerations, where the latter have to be transformed in common measurable units. The model allows to select one software architecture pattern from a given set of alternatives. Following [15], for each software architecture pattern i an architecture evaluation measure AEM_i is defined:

$$AEM_i = X \cdot OFM_i + (1 - X) \cdot SFM_i \tag{1}$$

where

AEM_i = architecture evaluation measure, $0 \leq AEM_i \leq 1$

OFM_i = objective factor measure, $0 \leq OFM_i \leq 1$ and $\sum_{i=1}^{n} OFM_i = 1$

SFM_i = subjective factor measure, $0 \leq SFM_i \leq 1$ and $\sum_{i=1}^{n} SFM_i = 1$

X = weight assigned to the objective factor, $0 \leq X \leq 1$

n = total number of software architecture patterns evaluated, $1 \leq i \leq n$

[4] The focus of the evaluation model is on the viewpoint of an integrator. The integrator takes into account purchase, licensing, set up, and maintenance costs for hardware and integration and maintenance costs for software. Development of software itself plays a secondary role, since the service or agent software has to be developed independent of the chosen architecture.

AEM_i is a measure between 0 and 1 for a particular software architecture pattern, where software architecture patterns with a higher measure score better than patterns with a lower measure. The measure depends to large extend on the choice of the weight X assigned to the objective factors OFM_i and the subjective factors SFM_i. This parameter can be used for sensitivity analysis.

Objective factors are quantified in terms of monetary units. In order to make them comparable to subjective factors, the objective factors have to be converted to a dimensionless index, i.e. an index with the dimension of one:

$$OFM_i = \frac{1}{OFC_i \cdot \sum_{i=1}^{n} \left(\frac{1}{OFC_i}\right)}, i = 1, 2, \ldots, n \tag{2}$$

where

$$OFC_i = \text{total objective factor costs for software architecture pattern } i$$

Brown and Gibson [7] ensure through three principles that the objective factor measure is compatible with the subjective factor measure: the software architecture pattern with the highest cost will have the minimum OFM_i, the relationship of OFC_i for each pattern relative to all other patterns is preserved, and the sum of all OFM_i is equal to 1.

The subjective factors can be grouped into a hierarchy of factors. A first level factors is an aggregation of a set of second levels factors. Within one first level factor the relative importance of a second level factor is rated by assigning a weight SSW_{k_j} to each of the second level factors. Similar the weight SFW_j specifies the relative importance of one first level factor to the other first level factors. Both factors weights depend on the *organizational context and the collaboration* for which the software architecture patterns are evaluated. The factor weights are independent of software architecture patterns, and can also be used for sensitivity analysis. The subjective factor measure SFM_i is defined as follows:

$$SFM_i = \sum_{j=1}^{m} \left(SFW_j \cdot \sum_{k=1}^{o_j} \left(SSW_{k_j} \cdot SAW_{ik_j} \right) \right) \tag{3}$$

$$SFW_j = \frac{SFW'_j}{\sum_{j=1}^{m} SFW'_j} \tag{4}$$

$$SSW_{k_j} = \frac{SSW'_{k_j}}{\sum_{k=1}^{o_j} SSW'_{k_j}} \tag{5}$$

$$SAW_{ik_j} = \frac{SAW'_{ik_j}}{\sum_{i=1}^{n} SAW'_{ik_j}} \tag{6}$$

where

SFW_j = normalized weight value of first level factor j

SFW'_j = weight of first level factor j to each first level factor

SSW_{k_j} = normalized weight value of 2nd level factor k_j for one 1st level factor j

SSW'_{k_j} = weight of second level factor k_j to all second level factors in first level factor j

SAW_{ik_j} = normalized rating of architecture variant i for subjective factor k_j

SAW'_{ik_j} = rating of architecture variant i for subjective factor k_j

m = total number of first level factors among the subjective factors

o_j = total number of second level factors in a specific first level factor j

All, the first level factor weight SFW_j, the second level factor weight SSW_{k_j}, and the architecture variant rating SAW_{ik_j} are normalized measures and sum up to one. Thus also the subjective factor measure SFM_i sums up to one and is represented in the same numerical scale as the objective factors. SFW_j, SSW_{k_j}, and SAW_{ik_j} are defined as follows:

4.2 Measuring qualitative factors

The part of the evaluation and decision model concerned with measuring qualitative factors is supposed to deal with two main challenges. First it has to provide concepts to evaluate software architecture patterns with respect to organizations' demands. Second the model has to provide means to support people using the model by rating factors and alternatives in order to achieve reasonable and consistent measurements throughout the evaluation process.

We use scenario-based evaluation for software architecture patterns, which is a good way to determine quality attributes of software architecture. The AHP [31] first decomposes a decision problem into a hierarchical network of factors and subfactors before it aggregates second level factors to first level factors. In scenario-based evaluation, first level factors are represented by quality attributes and second level factors are represented by scenario descriptions.

Since it is problematic to provide sensible scales for measuring the response value of our high level software architectural patterns, we make use of pairwise comparison (see AHP [31]) to rate the qualitative factors and the evaluated software architecture patterns. The decisions for the comparisons are made on the basis of which tactics the evaluated software architecture patterns support and the contingency factors influencing organizations and the collaboration.

Scenario-based ICT architecture evaluation Scenario-based ICT architecture evaluation is used to determine quality of software architecture. Hence desired architectural quality attributes are refined by general usage scenarios. These allow a detailed rating of how good quality attributes are supported by software architecture pattern. Quality attributes and scenarios descriptions are used to determine the qualitative factors measure.

Quality attributes Our evaluation model considers the strategic quality attributes modifiability, privacy, reusability and interoperability. For the quality attribute privacy we evaluate the privacy of corporate data and knowledge, which has to be exposed by the enterprises due to the applied software architecture pattern. We do not consider execution related topics like intrusion, denial of service attacks, etc. In the case of interoperability, which can be observed both at execution and build time, we only consider strategic issues like change and reuse of functionality or interaction protocols; we do not consider e.g. conversion of message data at runtime. Furthermore, the evaluation model addresses some more run-time related issues like efficiency and manageability of process execution.

Scenario descriptions The evaluation model is supposed to be suitable for a diversity of systems supporting businesses collaborations. Thus, general scenarios have to be developed, which can be applied to classes of systems rather than to one concrete system. Scenarios represent the characteristics of quality attributes and are used to determine how good quality attributes can be satisfied by systems realizing certain software architecture patterns. The following list gives an overview of the quality attributes (printed in boldface) and the associated scenarios defined for our evaluation and decision model.

- ***Modifiability***
  - *Scenario 1:* Modification of CBPs
  - *Scenario 2:* Change of partners in CBP
  - *Scenario 3:* Incremental development of CBPs
  - *Scenario 4:* Change of elementary services
  - *Scenario 5:* Development of CBP variants
- ***Privacy***
  - *Scenario 6:* Privacy of internal ESs related data
  - *Scenario 7:* Privacy of internal CBPs realizations
- ***Reusability***
  - *Scenario 8:* Reuse of CBPs
  - *Scenario 9:* Reuse of elementary services
- ***Interoperability***
  - *Scenario 10:* Change of CBP protocol specification
  - *Scenario 11:* Change of ES's interfaces
- ***Efficiency***
  - *Scenario 12:* Bottle-neck
  - *Scenario 13:* Security overhead
- ***Manageability***
  - *Scenario 14:* Versioning
  - *Scenario 15:* Monitoring

Table 1 depicts the description of the *'Modification of CBPs'* scenario. Descriptions of the other scenarios can be found in [28].

Scenario 1 – Modification of CBPs	
Source	Management
Stimulus	Due to the constant and rapid change in business existing CBPs have to be adapted to the new business models.
Environment	Design-time
Artifact	Cross-organizational business process
Response	The necessary changes in order to enact the new CBP affect a minimal number of existing modules. Necessary change of existing modules should have no side-effects on other processes (e.g CBPs).
Response Measure	Without broker: up to n ESs of the partners are affected Central broker: the central broker is affected Decentral broker: VPs of the respective partner(s) are affected

Table 1. Scenario 1 – Modification of CBPs

Factor decomposition and pairwise comparisons Factor decomposition and pairwise comparisons of our evaluation model are based on the Analytic Hierarchy Process (AHP) [31].

Factor decomposition Factor decomposition establishes a hierarchy of first level and second level factors cascading from the decision objective or goal. The hierarchy for our decision method is structured as follows (see Figure 7): At the top level one can find the overall goal to have the best *architecture quality*. At the first level contains quality attributes like *modifiability*, *privacy*, *reuse*, etc., which contribute to the quality of an architecture. The scenarios are used at the second level to give a more detailed description of how the quality attributes have to be established. At the bottom level we can find the architectural variants which have to support the scenarios.

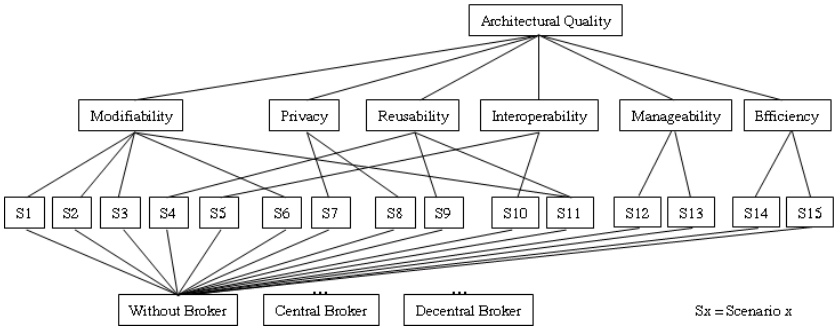

Fig. 7. AHP decomposition rree for CBP evaluation model

Pairwise comparisons AHP uses pairwise comparison for both determining the priority for the subjective factors and rating the architectural alternatives.

Weighting the subjective factors To determine the weights for subjective factors, i.e. which scenarios or quality attribute is more important than another, pairwise comparisons are conducted between the first-level factors and the second-level factors. Therefore the factors are arranged in a matrix a and the evaluators have to determine the ratings a_{ij} of the factors by pairwise comparisons. They use a scale to measure relative importance ranging from one to nine (one means that both factors are equally important; nine means that one factor is extremely more important than another). To calculate the ratios of the factors v_i, the entries of the matrix a_{ij} have to be normalized to $\overline{a_{ij}}$. Then the normalized matrix entries $\overline{a_{ij}}$ of each row are summed up and divided through the number factors, i.e. the average value of the normalized matrix entries for each row is determined.

$$v_i = \frac{\sum_{j=1}^{n} \overline{a_{ij}}}{n} = \frac{\sum_{j=1}^{n} \frac{a_{ij}}{\sum_{i=1}^{n} a_{ij}}}{n}$$

As a result, v_i is the weight for the respective SFW'_j or SSW'_{k_j} for the first and second level factors. It holds that $SFW'_j = SFW_j$ and $SSW'_{k_j} = SSW_{k_j}$ since the weights of the factors v_i are already normalized. The aggregation of the factor weights is achieved by multiplying the second level factor weight with the respective first level factor weight.

Rating the scenarios To rate the scenarios, our decision method applies a relative measurement, which based on a scale (see above) to express preference of one alternative over another. For example, one can say that to support a scenario under certain contingencies, alternative a_1 is strongly favored instead of alternative a_2. For each scenario an evaluation matrix is established, in which the alternatives are compared. To determine the rating of the alternatives (i.e. the priority vector), we apply the 'ideal mode' which should be used in cases where one alternative shall be chosen [30, 29]. The 'ideal mode' solves the rank reversal problem, where the number and kind of alternatives might influence the decision. The matrix is constructed analogous to the matrix for weighting the scenarios. Only the calculation of the priority vector's values differs, since we apply the 'ideal mode' and not the 'distributive mode'. One obtains the values of the priority vector in ideal mode v_i^{id} by dividing v_i by the maximal value of v: $v_i^{id} = \frac{v_i}{max(v_i)}$; v_i^{id} corresponds to the rating of the architecture variant SAW'_{ik_j}.

The measurement values of how good ICT coordination architectures support the scenarios is specific to organizational and collaboration context, i.e. the contingencies. It is possible that under certain contingencies one alternative is the best for supporting a scenario, while under different contingencies this alternative may be less appropriate to support the same scenario.

Rating the ICT architecture alternative To compare the ICT coordination architectures one needs to know how good these architectures support architecture quality attributes and scenarios. Therefore it is necessary to understand by

which means an architect influences the quality attributes of an architecture. As described in [1], software architects use so-called tactics to achieve quality attributes (see Section 2.5).

In the case of *scenario 1* the architect applies tactics that reduce the number of modules and processes (*response of scenario 1*) that are affected by changes to processes (*stimulus of scenario 1*). Through the *maintenance of semantic coherence* the architect ensures that the responsibilities among the services in a CBP work together without excessive reliance on each other. To *anticipate expected changes* reduces the services that need to be modified in case of certain changes. *Generalized services* allow to compute a broader range of functions based on the same input. An architect can apply these three tactics to CBP architectures by using the patterns *abstraction*, *loose coupling*, and *orchestration*.

With this information it is, in general, possible to decide whether one architecture variant supports a scenario better than another one. Having a look at *scenario 1* (cp. Table 1), the decentral broker architecture incorporates the patterns *abstraction*, *loose coupling*, and *orchestration* for *CBPs*, which is the artifact of the scenario description. Thus it realizes the tactics *maintain semantic coherence*, *anticipate expected changes*, and *generalize module*. The architecture without broker instead, realizes none of these patterns and tactics for the artifact (CBPs) of scenario 1. Thus we can infer that the decentral broker architecture better supports scenario 1 than the architecture without broker. The remaining question is, how contingencies influence the ratings and the distance between the ratings of the evaluated architectures.

Contingencies In our decision method we consider contingencies within the collaboration network (internal contingencies) and outside the collaboration network (external contingencies). Internal contingencies characterize the collaboration model and the organizations participating in the collaborations. These are: the *collaboration topology*, that takes into account the distribution of influence and power among the partners; the complexity and specificity of the *products* developed by the collaborating organizations; the *service flow* that is characterized by the amount of data and the number of messages exchanged; aspects related with the *process* itself like length of the process or the estimated number of process instance during execution. External contingencies are external factors that highly influence organizations' decision and strategies, and therefore impact also the choice of an ICT coordination architecture: *standardization* considers the existence of industry-specific, national, or international standards; *maturity* takes the existence of commonly accepted processes, protocols, etc., into account; *business semantics* considers the availability of standards and their maturity with regard to defining semantics of a specific domain; *legislation* comprises the regulations which impose special requirements regarding security, monitoring, and other aspects of the collaboration. [22]

If we assume for example a high degree of standardization to rate scenario 1, the decentral broker architecture is not much better than or even equal to the architecture without broker. Standardized parts of the CBP and the ESs can be reused and combined in arbitrary ways adapting to the change in business

(stimulus of scenario 1). Necessary changes affect about the same number of modules (cp. response and response measure of scenario 1 in Table 1) in both coordination architectures.

Of course there exist other contingencies, which are also relevant for the decision about an ICT coordination architecture. For example, the dynamics of the collaboration (internal contingency) and the industry dynamics (external contingency) both address the aspect of change. Since change is already covered by the scenario descriptions, this aspect has to be considered by weighing scenarios and quality attributes. Change is not addressed a second time in rating the scenarios.

4.3 Measuring quantitative factors

In the decision method quantitative factors are evaluated in monetary terms on the basis of the discounted cash flow approach.[5] The discounted present value of the future cash flows FV_i^D, which corresponds to the objective factor measure OFC_i for a software architecture pattern i, is defined as follows:

$$OFC_i = \sum_{j=1}^{m} FV_{i_j}^D = \sum_{t=0}^{N-1} \frac{FV_{i_{j_t}}}{(1+d)^t} \tag{7}$$

where

$FV_{i_j}^D$ = discounted present value of the future cash flow (FV) for factor j
$FV_{i_{j_t}}$ = nominal value of a cash flow amount in a future period t for factor j
d = discount rate
N = number of discounting periods
m = total number of objective factors

The decision model considers costs for software (*purchasing costs* and *annual licences*), hardware (*purchasing costs* and *annual leasing fees*) and labor (*costs to set up the systems*, *maintenance costs*, and *costs to develop and deploy new and modified processes*).

5 Applying the evaluation method

As described in the introduction, companies organize themselves into global networks and outsource those activities that can be performed quicker, more effectively, or at lower cost, by others [34]. However, outsourcing and interacting in global networks also increases overhead costs for collaboration, coordination, and intermediation. One approach to describe the influence of organizational structure on these overhead costs is the transaction cost model [37, 38]. In today's

[5] The description of the quantitative factors is quite short, since we focus on the qualitative factors and contingencies in this paper.

economies, transactions for example make up more than 30% of the total costs of an automobile [36]. Transaction costs heavily depend on the capabilities of business systems to keep up with constantly evolving business relationships and cross-organizational value chains. However, in the comparison to transaction costs, IT costs are much less than transaction costs (in the Automotive example this is about 6% of the overall costs [36]).

In this section, we apply the evaluation method to two scenarios: a virtual enterprise scenario (Section 5.1) and to a scenario with collaborating SMEs (Section 5.2). In doing so, the goal is to identify the collaboration architecture which best supports the cross-organizational value chain and helps to reduce transaction costs. The trade-off between reducing transaction costs (qualitative factors) and reducing of IT cost (quantitative factors) through the choice of a collaboration architecture is discussed in a sensitivity analysis.

5.1 Virtual enterprise scenario

This scenario deals with virtual enterprises that collaborate in big, long-running CBPs (approx. 90 processing steps). The OEM and the big first-tier suppliers introduced in the automotive scenario in Section 1 together form a virtual enterprise, which builds a temporary network of independent companies, suppliers, customers. They are linked by information technology to share costs, skills, and access to one another's markets. The services the partners provide to the CBP are to 50% legacy applications, which will be replaced within the next five years. The services, their interfaces, and data types are not standardized, so that interoperability is an important issue. About 30% of the CBP are standardized and it may be necessary to provide variants of the CBP. The privacy of the enterprises' services is only medium important, since the enterprises make their profit through economy of scale. Hence, they also participate with their elementary services in other CBPs.

Determining the qualitative measure

Weighting the subjective factors To determine the weight of the quality attributes and the scenarios pairwise comparison are applied like described in Section 4.2.

	mod.	pri.	reuse	int.	eff.	man.	v_i
modifiability	1	7	3	$\frac{1}{3}$	3	3	0.21
privacy	$\frac{1}{7}$	1	$\frac{1}{4}$	$\frac{1}{9}$	$\frac{1}{5}$	$\frac{1}{5}$	0.03
reuse	$\frac{1}{3}$	4	1	$\frac{1}{5}$	1	1	0.10
interoperability	3	9	5	1	5	5	0.45
efficiency	$\frac{1}{3}$	5	1	$\frac{1}{5}$	1	1	0.10
managability	$\frac{1}{3}$	5	1	$\frac{1}{5}$	1	1	0.10

Table 2. Priority comparison matrix for the first level factors

Table 2 depicts the weighting of the first level factors, i.e. the quality attributes, for the virtual enterprise scenario. Modifiability is considered more important than privacy and reuse but less important than interoperability. The column of the priority vector v_i depicts the weighting of the quality attributes.

modifiability	sc.1	sc.2	sc.3	sc.6	sc.11	v_i
scenario 1	1	3	7	$\frac{1}{5}$	$\frac{1}{3}$	0.14
scenario 2	$\frac{1}{3}$	1	5	$\frac{1}{5}$	$\frac{1}{5}$	0.08
scenario 3	$\frac{1}{5}$	$\frac{1}{7}$	1	$\frac{1}{9}$	$\frac{1}{9}$	0.03
scenario 6	5	5	9	1	3	0.47
scenario 11	3	5	9	$\frac{1}{3}$	1	0.27

Table 3. Priority comparison matrix for the second level factor modifiability

Table 3 depicts the weighting of the scenarios are used to describe the modifiability attribute in the virtual enterprise scenario. The scenarios are analogously compared as the other quality attributes in Table 2. The column of the priority vector v_i depicts the weighting of the scenarios.

Rating the Scenarios The scenarios are rated by pairwise comparing the architecture alternatives. The decisions are based on how good the architectures support the scenarios via tactics and patterns). The rating, i.e. the values decision, also depend on the characteristics of the contingency factors of the application scenario for which the evaluation is performed.

scenario 1	Wo-Br.	Cen-Br.	Dec-Br.	v_i^{id}
Wo-Br.	1	$\frac{1}{7}$	$\frac{1}{7}$	0.14
Cen-Br.	7	1	1	1.00
Dec-Br.	7	1	1	1.00

Table 4. Rating scenario 1

Table 4 depicts the rating matrix for scenario 1. As described in Section 4.2 the central broker alternative supports scenario 1 better than the brokerless alternative. Relevant contingencies for scenario 1 are the grade of standardization and the maturity of the CPB and the services. Since both contingencies are rather low in the virtual enterprise scenario, the architectural quality is important for the support of this scenario, which leads to the comparison value 7 between the central broker and without broker architecture. The central and decentral broker architecture are rated equally important with the value 1. The column of the priority vector v_i^{id} depicts the weighting of the scenarios.

Overall Subjective Measure The overall subjective measure is computed on the basis of the factor weights and the scenario ratings. Table 5 depicts the relevant data. In row two one can find the weighting of the quality attributes from Table

2. The weighting of the scenarios that describe the quality attributes are specified in row four. The scenario ratings can be found in the columns of the respective scenarios. For example the rating, i.e. priority vector values v_i^{id}, for scenario 1 can be found in column 2 row 5-7. The overall subjective measure is calculated with the formula (3) and can be found in the last column.

	Modifiability					Sec.		Reuse			Int.op.		Eff.		Man.		SFM_i^{id}	SFM_i
	0.21					0.03		0.10			0.45		0.10		0.10			
	S1	S2	S3	S6	S11	S7	S8	S4	S9	S11	S5	S10	S12	S13	S14	S15		
	0.14	0.08	0.03	0.47	0.27	0.17	0.83	0.43	0.43	0.14	0.17	0.83	0.50	0.50	0.50	0.50		
Wo-Br.	0.14	0.11	0.08	0.16	0.08	0.30	0.20	0.30	0.17	0.08	0.14	0.17	1.00	0.17	0.12	0.11	0.202	0.121
Cen-Br.	1.00	0.44	0.30	0.46	0.30	0.11	0.20	0.12	0.59	0.30	1.00	0.41	0.33	1.00	1.00	1.00	0.545	0.327
Dec-Br.	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.41	0.60	0.44	0.920	0.552

Table 5. Overall subjective measure

Determining the quantitative measure

Overall Objective Measure The objective measure is calculated on the basis of the cash flow of the costs for software, labor, hardware. For the virtual enterprise scenario with four collaborating enterprises we have estimated the following costs. It is important to understand, that the scale (euro, dollar, etc.) is not important for the overall objective measure, since the scale is transformed into an dimensionless index. In Table 6 one can see that for the architecture without broker 5075 thousand cost units were estimated (OFC_i). The overall objective measure OFM_i can be found in the last column.

	Software	Hardware	Labour	OFC_i	OFM_i
Wo-Br.	45K	75K	4955K	5075K	0.127
Cen-Br.	69K	95K	1200K	1364K	0.471
Dec-Br.	118K	135K	1367K	1620K	0.399

Table 6. Overall subjective measure

Sensitivity analysis and interpretation The architectural evaluation measure AEM_i for each architecture variant is determined on the basis on the objective factor measure OFM_i and the subjective factor measure SFM_i (see formula (1)). The measure depends on the weight X assigned to the objective and subjective factor. This weight lends itself also for sensitivity analysis.

Figure 8 depicts the sensitivity analysis chart for the virtual enterprise scenario. The x-axis represents the importance of the objective factors measure

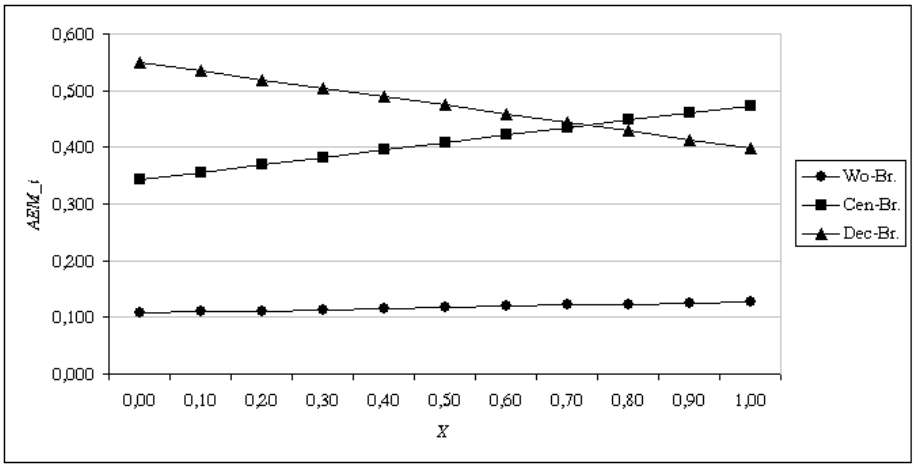

Fig. 8. Sensitivity analysis chart

and the y-axis the architecture evaluation measure for the respective architecture variant.

On the basis of this evaluation result we can conclude, that either the central broker or the decentral broker architecture variant should be selected. The variant without broker gets significantly lower rating values for all X than the other ones. The decentral broker architecture scores better for the qualitative measurement (especially for $X = 0$), while the central broker architecture is better in terms of IT costs. A feasible estimation of X is to consider the relationship between the percentage of transaction costs and IT costs of the total costs. In the automotive industry IT costs (6%) are low in comparison to the transaction costs (30%) (cp. [36]). This leads to an estimation of $X \approx 0.2$ for the virtual enterprise scenario applied to the automotive industry. Thus, we would suggest to select the decentral broker architecture in the virtual enterprise scenario. Even if transaction costs and IT costs got equally important ($X = 0.5$), the architecture evaluation measure of the decentral broker variant would be still be a bit better than the central broker variant.

5.2 SME scenario

This scenario represents the CBPs between the second-tier (or even third- and fourth-tier suppliers) of the automotive scenario from Section 1. The second-tier suppliers are SMEs that manufacture parts, which can be largely standardized and can be reused in many cars or other application domains. The SMEs produce for example screws, fuses, circuit boards, etc.. They support rather short processes with approx. 20 processing steps. The specificity of the service is low. Smaller and equal partners (SMEs) frequently join and leave the collaborations and most SMEs also participate in other similar collaborations. Participating

partners have similar interfaces, data types, etc., and the services and CBPs are de-facto standardized (e.g. already formulated in ebXML). Hence, interoperability is not so an important issue to these organizations. Also changes to the existing CBPs are rare (up to three times a year). However, about 50% of the service offer by the SMEs are legacy applications, which will be partially replaced within the next five years.

Determining the qualitative measure The overall subjective measure can be found in Table 7.

	Modifiability 0.16					Sec. 0.04		Reuse 0.25			Int.op. 0.06		Eff. 0.25		Man. 0.25		SFM_i^{id}	SFM_i
	S1	S2	S3	S6	S11	S7	S8	S4	S9	S11	S5	S10	S12	S13	S14	S15		
	0.05	0.59	0.05	0.21	0.11	0.50	0.50	0.45	0.45	0.09	0.20	0.80	0.83	0.17	0.17	0.83		
Wo-Br.	0.39	0.36	0.50	0.20	0.40	1.00	1.00	1.00	0.17	0.40	0.17	0.40	1.00	0.50	0.30	0.17	0.529	0.293
Cen-Br.	1.00	0.50	1.00	1.00	1.00	0.64	1.00	1.00	0.30	0.41	0.64	0.41	0.64	0.12	1.00	1.00	0.589	0.326
Dec-Br.	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.55	1.00	1.00	1.00	1.00	0.40	1.00	0.50	1.00	0.688	0.381

Table 7. Overall subjective measure

Determining the quantitative measure The overall objective measure can be found in Table 8.

	Software	Hardware	Labour	OFC_i	OFM_i
Wo-Br.	100K	100K	100K	300K	0.453
Cen-Br.	124K	120K	195K	439K	0.310
Dec-Br.	197K	180v	198K	575K	0.237

Table 8. Overall subjective measure

Sensitivity analysis and interpretation Figure 9 depicts the sensitivity analysis chart for the SME. One can clearly see how the contingencies standardization and short processes influence the architecture evaluation measure. Although the partners in the collaboration frequently change the architecture variant without broker scores very well. For most X, the brokerless architecture has the highest evaluation measure and even for low X its measure is hardly lower than the measure for the broker architecture. However, if contingencies change, like new monitoring requirements from the government, the intersection point of the curves would be at a higher X (it would move to the right). This would make the broker architectures more interesting to realize the SMEs scenario.

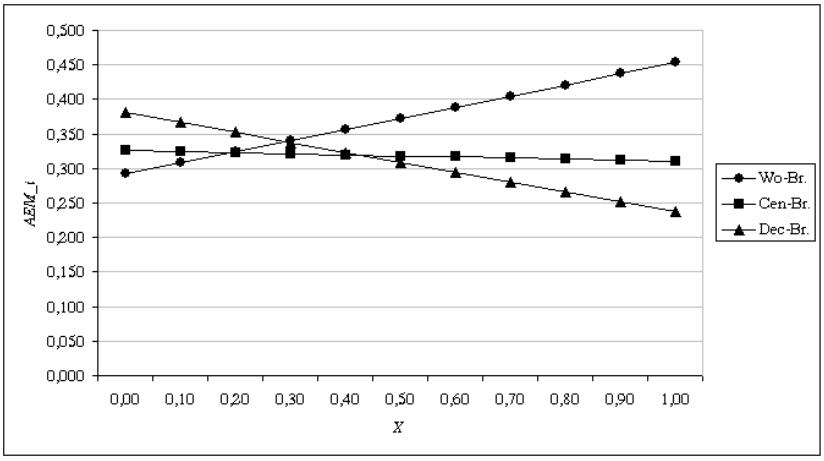

Fig. 9. Sensitivity analysis chart

6 Discussion and outlook

The contribution of the work reported in this paper is fourfold: First, we identified three important architectural patterns for CBP enactment: brokerless architecture, decentral broker architecture, and central broker architecture. Second, we proposed a decision method for architecture selection based on the analytic hierarchy process (AHP, see [31]). This method targets ICT architects and promises a systematic way to evaluate and compare ICT level architecture variants for a certain application scenario, based on pairwise comparison of alternatives. Third, we showed the applicability of our method by applying it to two application scenarios with differing characteristics: A virtual enterprise scenario (comparable to the relationship between the OEM and the selected first-tier suppliers discussed in Section 1) and an SME network scenario (similar to the second-tier network part of the example in Section 1). Finally, we investigated the robustness of the decision method by performing a sensitivity analysis.

An area for future work is the examination and deeper evaluation of the decision method. One aspect concerns the choice of making pairwise comparisons between alternatives as described in Section 4. Our experience so far indicates that pairwise comparisons reduce the amount of information that is necessary for decisions. Since people can only deal with information involving simultaneously a small number of facts (seven plus or minus two) [25], pairwise comparisons help evaluators to make better judgements compared to methods where more information needs to be considered. Though pairwise comparisons require more complex calculations than other rating approaches, they promise to provide more exact results. The AHP method involves also redundant comparisons to improve validity, recognizing that participants may be uncertain or make poor judgements in some of the comparisons. Further investigation are needed to achieve a

more fundamental understanding of the trade-offs involved in redundant pairwise comparisons and possible alternatives.

A second area concerns the question how decision methods as the one described in this paper can be built into existing enterprise modeling frameworks and model-driven IDEs, to support process modelers and ICT architects in their task of creating and managing executable CBP specifications from business level models. Also, more fine-grained models and extensions of our decision method need to be developed to support this process down to the platform-specific and code levels.

References

1. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. Addison-Wesley (2003)
2. Bass, L., John, B.E.: Linking Usability to Software Architecture Patterns Through General Scenarios. Journal of Systems and Software **66** (2003) 187–197
3. Bennett, D.W.: Designing Hard Software - The Essential Task. Prentice Hall (1997)
4. Bézivin, J.: On the unification power of models. Software and System Modeling **4** (2005) 171–188
5. Birman, A., Ritsko, J.: Preface to Service-Oriented Architecture. IBM Systems Journal **44** (2005) 651–652
6. Booch, G., Brown, A., Iyengar, S., Rumbaugh, J., Selic, B.: An MDA Manifesto. MDA Journal (2004)
7. Brown, P.A., Gibson, D.F.: A quantified model for facility site selection application to multiplant location problem. AIIE transactions: industrial engineering research and development **4** (1972) 1–10
8. Clements, P., Kazman, R., Klein, M.: Evaluating Software Architecture. Addison-Wesley (2002)
9. Dolan, T.J.: Architecture assessment of Information-System Families. PhD thesis, Technische Universiteit Eindhoven (2001)
10. Donaldson, L.: The Contingency Theory of Organizations. SAGE Publications, Inc (2001)
11. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall International (2005)
12. Erl, T.: Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services. Prentice Hall International (2004)
13. Gamma, E., Helm, R., Johnson, R.E.: Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley Longman (1995)
14. Garcia-Camino, A., Noriega, P., Rodriguez-Aguilar, J.: Implementing Norms in Electronic Institutions. In: Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05), ACM Press (2005) 667–673
15. Ghandforoush, P., Huang, P.Y., Taylor, B.W.: A mulit-criteria decision model for the selection of a computerized manufacturing control system. International Journal of Production Research **23** (1985) 117–128
16. Greenfield, J., Short, K., Cook, S., Kent, S.: Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. Wiley Publishing Inc. (2004)

17. Hailpern, B., Tarr, P.: Model-driven development: The good, the bad, and the ugly. IBM Systems Journal **45** (2006) 451–461
18. Jennings, N.R., Sycara, K., Wooldridge, M.: A Roadmap of Agent Research and Development. Journal of Autonomous Agents and Multi-Agent Systems **1** (1998) 7–38
19. Jennings, N.R., Faratin, P., Norman, T.J., O'Brien, P., Odgers, B.: Autonomous Agents for Business Process Management. Int. Journal of Applied Artificial Intelligence **14** (2000) 145–189
20. Jennings, N.R., Faratin, P., Norman, T.J., O'Brien, P., Odgers, B., Alty, J.L.: Implementing a Business Process Management System using ADEPT: A Real-World Case Study. Int. Journal of Applied Artificial Intelligence **14** (2000) 421–465
21. Larsen, G.: Model-driven development: Assets and reuse. IBM Systems Journal **45** (2006) 541–553
22. Legner, C., Wende, K.: Towards an Excellence Framework for Business Interoperability. In: 19th Bled eConference "eValues", Slovenia. (2006)
23. Leymann, F., Roller, D., Schmidt, M.T.: Web services and business process management. IBM Systems Journal **41** (2002) 198–211
24. Malone, T.W.: Modeling Coordination in Organizations and Markets. Management Science **33** (1987) 1317–1332
25. Miller, G.A.: The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information. The Psychological Review **63** (1956) 81–97
26. OASIS: Web Services Business Process Execution Language Version 2.0. wsbpel-primer (2007)
27. OMG: Business Process Definition MetaModel (BPDM), final submission. bmi/2006-11-03 (2006)
28. Roser, S.: Designing and Enacting Cross-organisational Business Processes: A Model-driven, Ontology-based Approach. PhD thesis, University of Augsburg (2008, forthcoming)
29. Saaty, T.L.: Decision Making for Leaders. 3rd edn. RWS Publications (1999)
30. Saaty, T.L.: How to make a decision: The Analytic Hierarchy Process. Interfaces **24** (1994) 19–43
31. Saaty, T.L.: The Analytic Hierarchy Process. McGraw-Hill, New York (1980)
32. Schmidt, D.C.: Guest Editor's Introduction: Model-Driven Engineering. Computer **39** (2006) 25–31
33. Singh, M., Huhns, M.: Service Oriented Computing: Semantics, Processes, Agents. John Wiley & Sons, Chichester, West Sussex, UK (2005)
34. Snow, C.C., Miles, R.E., Coleman, H.J.: Managing 21st Century Network Organizations. Organizational Dynamics **20** (1992) 5–20
35. Stäber, F., Sobrito, G., Müller, J., Bartlang, U., Friese, T.: Interoperability challenges and solutions in Automotive Collaborative Product Development. In Gonçalves, R., Müller, J., Mertins, K., Zelm, M., eds.: Enterprise Interoperability II: New Challenges and Approaches. Springer-Verlag (2007) 709–720
36. Strassmann, P.A.: Is Outsourcing Profitable? Lecture at George Mason University (2006)
37. Williamson, O.E.: Markets and Hierarchies: Analysis and Antitrust Implications. Free Press (1975)
38. Williamson, O.E.: Transaction Cost Economics. Handbook of Industrial Organization **1** (1989) 135–182