An evaluation and decision method for ICT architectures for cross-organizational business process coordination

Stephan Roser · Jörg P. Müller · Bernhard Bauer

Abstract Our work aims at providing support for the decision-making processes involved in the model-driven development of information technology (IT) solutions for cross-organizational business process (CBP) coordination and automation. The objective of the work described in this paper is to provide enterprise IT architects with an evaluation and decision model that enables the principled assessment and selection of an effective IT architecture paradigm (e.g. central broker, federated brokers, peer-to-peer) for a given cross-organizational business process coordination task. Our approach follows the principles of design science; the contribution of this paper is threefold: First, we present three common architectural patterns for (service-oriented) CBP coordination. Second, the core contribution is established by a new method for decision support suitable for IT architects to derive and evaluate an appropriate architecture paradigm for a given use case or application domain. The method is accompanied by a set of representative scenario descriptions that allow the evaluation and selection of appropriate IT system coordination architecture paradigms for CBP enactment, as well as a set of guidelines for how different contingencies influence IT system coordination architecture.

Keywords Design science · Software architecture · Evaluation · Collaborative business processes · Service orientation · Model driven engineering

S. Roser
Senacor Technologies AG, Wieseneckstr. 26, 90571 Schwaig b. Nürnberg, Germany
e-mail: stephan.roser@senacor.com

J. P. Müller
Department of Informatics, Clausthal University of Technology, Clausthal-Zellerfeld, Germany
e-mail: joerg.mueller@tu-clausthal.de

B. Bauer (✉)
Programming Distributed Systems Lab, University of Augsburg, Augsburg, Germany
e-mail: bauer@ds-lab.org

1 Introduction

Today, companies are urged to adapt to market pressures and competitors' innovations with increasing speed. They globally search for opportunities and resources and perform only those functions for which the company has expert skills. Companies are organized in global networks and outsource those activities that can be performed more quickly and effectively or at lower costs, by others Snow et al. (1992). In this context, transaction costs (Williamson 1975, 1989) are a crucial performance indicator. Transaction costs are those costs incurred by indirect production expenses through imperfect economic exchange (Wallis and Douglas 1986, p. 97). Transaction costs account for more than 30% of the total costs of an automobile and about 50% of the total costs of a Logitech mouse (Strassmann 2006). Business systems govern nearly all forms of transactions by facilitating business relationships and value chains. Thus, in modern economies low transaction costs heavily depend on the capabilities of business systems to keep up with constantly evolving business relationships and cross-organizational value chains.

Enterprise Application Integration (EAI) projects serve this need of organizations to form networks and work together by coupling diverse Information and Communication Technology (ICT) systems. Recently, EAI has been extended beyond the boundaries of individual organizations, thus supporting, coordinating, or automating cross-organizational business processes (CBPs).[1] CBP coordination solutions face, like the organizations themselves, the challenge to adapt their processes with increasing speed and to respond quickly to changing requirements.

A new generation of CBP coordination solutions has been developed under the service-oriented paradigm. 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 a CBP coordination solution by itself. While it introduces the concept of *services* to establish a platform-independent model with various integration architectures, supporting ICT architects to evaluate and select architecture paradigms and topologies, which are appropriate for concrete business requirements, is still an open problem. Existing approaches (Bass et al. 2003; Bass and John 2003; Clements et al. 2002) are designed to evaluate the architectures of concrete ICT systems. Their measuring methods do not yield sensible results when they are applied to high-level ICT architectures. Further, the above-mentioned approaches do not take into account how the various aspects of the environment of organizations influence the assessment of (and the decisions taken in) the evaluation process.

Thus we have rapidly changing processes, transaction cost to be considered and CBP and SOA as state of the art for the realization of such systems. But, given a specific CBP scenario, how can we obtain a "good" SOA architecture from a cost as well as technological point of view? The specific aim motivating this paper is to provide the ICT architect with an evaluation and decision model that enables the

[1] It is worth noting here that CBPs not only occur between different enterprises, but also in intra-enterprise settings, for instance between different members of a global group of companies, or between different business units of a geographically distributed enterprise.

principled assessment and selection of "the right" ICT architecture for a given CBP scenario. In this paper, we present a solution to this problem, following a model-driven engineering approach; we define a decision method to select the most appropriate from a set of typical architecture patterns. We particularly address the decision tasks arising for IT architects when mapping a set of computation-independent enterprise models to a platform-independent CBP coordination model. At this stage of the model-driven development process, different alternative topologies may need to be considered for mapping the enterprise model to an ICT architecture model: for instance, a centralized message broker topology, a heterogeneous "multi-broker" topology, or a decentralized peer-to-peer topology. Thus, our work closes a gap and differentiates itself from other research on architecture evaluation (see Sect. 2) by focusing on the decision-making processes involved in the development of ICT solutions for CBP coordination and automation.

Our work follows a design science approach (Hevner et al. 2004). Design science artifacts are classified in constructs, models, methods, and instantiations. *Constructs* define the language used to describe and communicate problems and solutions. *Models* use constructs in order to represent problems and solution spaces. *Methods* in design science define the process how problems are solved (e.g., for searching the solution space). They include exact, algorithmic methods as well as informal, textual descriptions of best practices. Finally, the purpose of *instantiations* is to show *that* (and *how*) constructs, models, and methods can be used to design a solution (e.g., by illustrating their feasibility in case studies).

After an overview of the background and state-of-the-art in Sect. 2, we present the basic *constructs and models* used throughout this work. Here, the first contribution (as depicted in Sect. 3) are three common architectural patterns for (service-oriented) CBP coordination. The second, core contribution of this paper (see Sect. 4) is in the area of design science *methods*:

- We present a method for decision support suitable for ICT architects to derive and evaluate an appropriate architecture paradigm for a given use case or application domain. The decision support method combines the Analytic Hierarchy Process (AHP) (Saaty 1980) with scenario-based architecture evaluation techniques (Bass et al. 2003; Clements et al. 2002).
- A set of representative scenario descriptions that allow the evaluation and selection of appropriate ICT system coordination architecture paradigms for CBP enactment.
- Based on our experiences in model-driven CBP modeling and enactment, we propose a set of guidelines for how different contingencies (Donaldson 2001), i.e. internal and external influence factors that determine the effectiveness of an organization, influence ICT system coordination architecture.

Finally, Sect. 5 establishes the third contribution of this paper: We provide *instantiations* of our framework by applying the decision support method, the scenario descriptions for CBP enactment, and the guidelines to a set of application scenarios with differing characteristics. The paper ends with concluding remarks and an outlook to future research issues in Sect. 6.

2 Background

In this section, we give an overview of the technologies used in the following. Since we assume a service-oriented CBP realization, in Sect. 2.1 we first introduce the service-oriented paradigm; subsequently, in Sect. 2.2, we elaborate on CBPs. Since the main contribution of this paper is the decision method for service-oriented CBP architectures, in Sect. 2.3, we give a quick overview on architecture evaluation and on decision making approaches, namely analytic hierarchy processes and contingency theory. Section 2.4 puts these concepts in the context of our work; the integration of these methods and approaches will be detailed in Sect. 4 onwards.

2.1 Service-oriented application integration

Over the past decade, a new generation of integration solutions has been developed under the labels of Service-Oriented Architecture (SOA) or Service-Oriented Computing (SOC). The goal is to support a 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. Usually SOA starts at the business level and is afterwards refined to e.g. Web Services. Starting with business services offered to e.g. a customer, these services are divided up into IT- and non-IT services to automate the process as much as possible.

Technically a service can be defined *"as a well-defined, self-contained function that does not depend on the context or state of other services"* (Birman and Ritsko 2005). In ICT architectures based on service-orientation one can distinguish various notions of basic service models (from Erl 2005):

- Application Service: A service that contains logic derived from a solution or technology platform.
- Business Service: A service that contains business logic.
- Hybrid Service: A service that contains both business and application logic. Most services created as part of traditional distributed solutions fall into this category.
- Controller Service: A service that composes others.
- Process Service: A service that represents a business process as implemented by an orchestration platform and described by a process definition.

Enterprise application integration projects serve the need of organizations to form and work together in networks by coupling arbitrary ICT systems. The main objective of most integration initiatives is to achieve a new level of interoperability while minimizing the impact on existing ICT environments.

According to Erl (2004) service-oriented integration means:

- avoiding the creation of a fragmented environment through the introduction of business logic that resides outside of established application boundaries.
- avoiding tightly bound integration channels between applications that are easily broken if either application is modified.
- minimizing redevelopment of applications affected by the integration.

Like the organizations themselves, EAI solutions face the challenge to adapt their processes with increasing speed and to respond quickly to changing requirements. Integration has been accomplished through a variety of point-to-point integration models, most commonly facilitated by broker and orchestration components.

2.2 Cross-organizational business processes

2.2.1 Business processes

People with different backgrounds, i.e. business or IT, that speak about processes do not always mean the same. IT people often refer to the term process in the context of process execution and workflows. Business people use process to describe procedures within and between organizations, that are of coarse granularity at a high level of abstraction and can often not be executed (directly) by workflow engines. Hammer and Champy (1993) "define a business process as a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer". The term business process is used by both camps. However, business processes often include manual activities and can be related to every kind of resource. A variety of languages for describing workflows and process executions like Web Services Business Process Execution Language (WS-BPEL) (IBM et al. 2003; OASIS 2007), Business Process Modeling Language (BPML) (Arkin 2002), Web Service Choreography Interface (WCSI) (W3C 2002), Yet Another Workflow Language (YAWL) (van der Aalst and ter Hofstede 2005), or XML Process Definition Language (XPDL) (WfMC 2005) has been developed. Other languages like Business Process Modeling Notation (BPMN) (OMG 2006), UML 2.0. Activity Diagrams (OMG 2007), or Event-driven Process Chains (EPCs) (Klein et al. 2004) are used to represent processes at higher level of abstraction.

2.2.2 Orchestration and 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 (OMG 2006). 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 (Erl 2005).

2.2.3 Cross-organizational business processes

In order to coordinate inter-organizational workflows, Liu and Shen introduced the concept of views, as they are used in database systems, to provide abstract

information about internal processes. In Liu and Shen (2001) they extend their work to Cross-organizational Business Processes (CBPs). Chiu et al. introduce workflow views to control visibility of internal processes and to enable interoperability of e-services, focusing on combining views of different partners to composite CBPs. Schulz et al. use the concept of views, and formalize the dependencies between private processes, process views, and CBPs (Schulz and Orlowska 2004). Adopting the general approach of (Schulz and Orlowska 2004), we distinguish between Private Processes (PPs), View Processes (VPs), and CBPs (according to Lippe et al. 2005):[2]

- Private Processes (PPs) are internal to an organization. They contain data not to be revealed by default. Views on processes provide an abstraction of PPs, which is sufficient to coordinate internal actions with activities of external business partner(s) (Schulz and Orlowska 2004). A particular interaction may require the involved partners to adapt for the purpose of the communication. This adaptation may not necessarily be reflected in the partners' private (internal) business processes without impairing their ability to interact with other partners in a different context.
- View Processes (VPs) combine PPs and form an abstract level that enables companies to hide critical information from unauthorized partners. The VP connects the PP with the abstract process an organization provides to a CBP. Based on one PP, different views can be generated, which reflect the specific requirements of different interactions.
- Cross-organizational Business Processes (CBPs) define the interactions between two or more business entities. These interactions take place between the defined abstract processes and are defined as a sequence of message and/or other material input/output exchange. Using different views of the same internal processes, organizations are able to interact in different contexts without changing the internal process.

2.3 Architecture evaluation and decision methods

2.3.1 Architecture evaluation

Scenario-based ICT architecture evaluation is used to determine quality of software architecture. In architecture evaluation methods like ATAM, SAAM, or ARID (Bass et al. 2003; Clements et al. 2002), quality attributes are characterized by scenario descriptions. Most of these methods were developed at the Software Engineering Institute of the Carnegie Mellon University.

Quality attributes are part of the non-functional requirements and therefore properties of a system. They can be broadly grouped into two categories (Dolan 2001): qualities like performance, security, availability, interoperability, and usability are observable via execution (i.e. at *run-time*), whereas qualities like

[2] Note that the elementary primitives for process modeling are still executable, abstract, and collaborative processes.

extensibility, modifiability, portability, reusability, interoperability, etc. which are not observable via execution but at *build-time* (Bennett 1997).[3]

According to Bass et al. (2003), 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 (Bass and John 2003) are applicable to many software systems and have architectural implications; they establish sets of scenarios which are configured for 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 well a quality attribute or a scenario is supported by a software architecture pattern and to compare architecture patterns, it is crucial to understand how the choice of architecture influences quality attributes. According to Bass et al. (2003) 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 (Bass et al. 2003) 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. The following list provides architecture patterns, design patterns, and design principles used to realize the above described tactics (non-exclusive list compiled from (Bass et al. 2003; Erl 2004, 2005, and Gamma et al. 1995): *Wrapper*, *broker*, *abstraction*, *loose coupling*, and *orchestration*.

2.3.2 Analytic hierarchy process

The Analytic Hierarchy Process (AHP) (Saaty 1980) 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.

[3] The interoperability quality attribute comprises both aspects which are observable at run-time and aspects that are only observable at build-time.

In Saliu and Ruhe (2005) present an approach to assess software components which is based on the AHP. They evaluate the overall goal 'Difficulty of modification' via pairwise comparisons of software components. Their approach targets the support of release planning of component-based software. They use AHP in a fine-grained manner, i.e., the calculation of the overall rating is extended to explicitly consider the judgements of different individual experts. Also, they do not cover architectural issues, and—to the best our knowledge—they do not support the wide range of qualitative and quantitative criteria that are important for enterprise IT architecture evaluation. Also, the approach does not explore how qualitative architectural aspects can be elicited from scenario descriptions, and how different quantitative and qualitative aspects can be balanced.

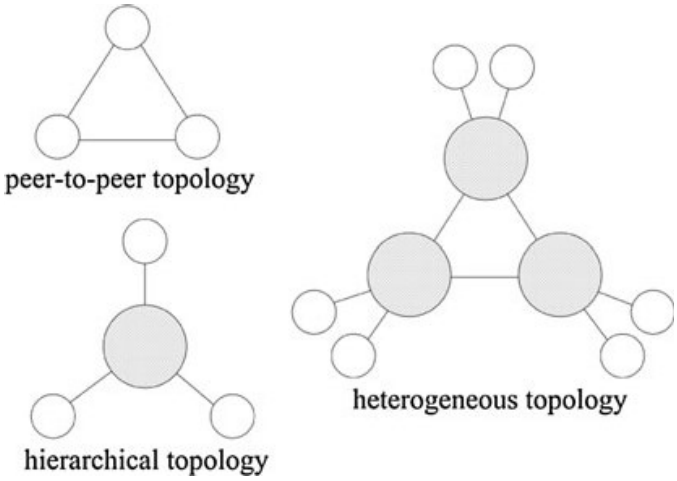
2.3.3 Contingency theory

Contingency theory for organizations (Donaldson 2001) is used to rationalize how the various aspects of organizational environments (called *contingency factors*) influence organization structure. It suggests, that there is no unique or best way to structure an organization; rather, 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"* [p. 1,10]. *"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"* [p. 4,10]. 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 rarely yield maximum performance of an ICT system for the overall business goals.

2.4 Summary

Many people and organizations participate in the construction of a large enterprise 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 architecture of software systems for CBPs can be derived from business level descriptions. We describe architecture variants and model transformations that are independent of functional attributes, since they can be applied to (nearly) any models describing CBPs. In the following,

Fig. 1 Coordination topologies



we examine three architecture variants for realizing CBPs in service-oriented software systems, how they can be derived from business level descriptions, and how the most suitable architecture can be derived for different contexts, thus supporting the enactment of high-level CBP specifications.

3 Software architectures for ICT system coordination

CBP coordination solutions can be categorized by their topology (see Fig. 1). In a purely decentralized peer-to-peer 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 a realization of orchestration. However, in many cases, as our experience in industrial projects has shown, a mixture of hierarchical and decentralized peer-to-peer topology, i.e. a heterogeneous topology, is used to realize complex multi-partner collaborations (Leymann et al. 2002).

These three abstract topologies for CBP enactment provide the basis for a continuum of coordination architectures. In the following we have a closer look at three concrete coordination architectures, which are examples of the three topologies, and how they can be applied to realize service-oriented solutions. These architectures are used to control the conversation flow between the participating organizations.

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 (ESs). In our descriptions we also assume without loss of generality, that the ESs are realized as process services, so that we can distinguish between executable and abstract processes. Nevertheless, ESs could be realized by arbitrary code fragments. An ES can only be a controller service with regard to the organizations' internal service composition, but not with regard to the collaboration process. CBPs represent the conversation flow and

message exchange between the organizations participating in the collaboration. We distinguish between the following architectural patterns:

- *Brokerless architecture*: A brokerless coordination architecture (see Fig. 2) can be used to realize the decentralized peer-to-peer topology, where messages are exchanged directly between the ESs of the participants as usual in a service-oriented world. Due to the mutual exchange of messages the ESs depend on each other. Control flow logic of CBPs is realized by the executable process of the participants' ESs. Changing the business protocol would result in changing multiple ESs, i.e. their executable processes. Further, the abstract process of the ESs are directly exposed to the collaboration space and therefore are directly accessible by entities outside enterprise boundaries.
- *Central broker architecture*: Fig. 3 depicts the central broker coordination architecture. Messages are no longer exchanged directly between the ESs, but over a central broker component, which is realized by a controller service. The controller service is a process that orchestrates the ESs 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 ESs from their collaborators.
- *Decentralized broker architecture*: The *decentralized broker architecture* introduces elements of the decentralized peer-to-peer topology into the hierarchical topology of the central broker architecture. It splits the single broker component into several controller processes jointly providing the broker functionality (see Fig. 4). Each organization provides one controller service, also called view process (VP) (cp. Sect. 2.2), which orchestrates the organization's internal ESs. Messages across organizational boundaries are only exchanged by the VPs, which encapsulate the ESs. In this architecture, elementary services can be seen as some type of private processes (PPs).

4 A method for evaluating ICT architecture applicability

This section presents an evaluation and decision method that serves IT architects to select appropriate ICT architectures for CBP enactment.

The evaluation method considers the trade-offs between coordination structures, which are implemented by the ICT system architectures, in terms of coordination costs and vulnerability costs (see Malone 1987). As illustrated in Fig. 5 the evaluation model distinguishes between quantitative factors, that are measurable by concrete figures (objective factors), and qualitative factors (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.

Vulnerability costs, which are "the unavoidable costs of a changed situation that are incurred before the organization can adapt to a new situation" (Malone 1987), are qualitative factors in the evaluation model.

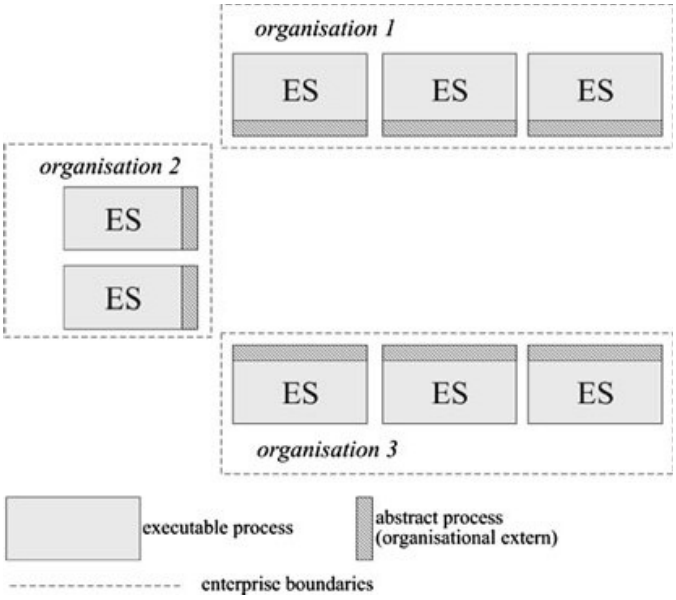


Fig. 2 Brokerless architecture

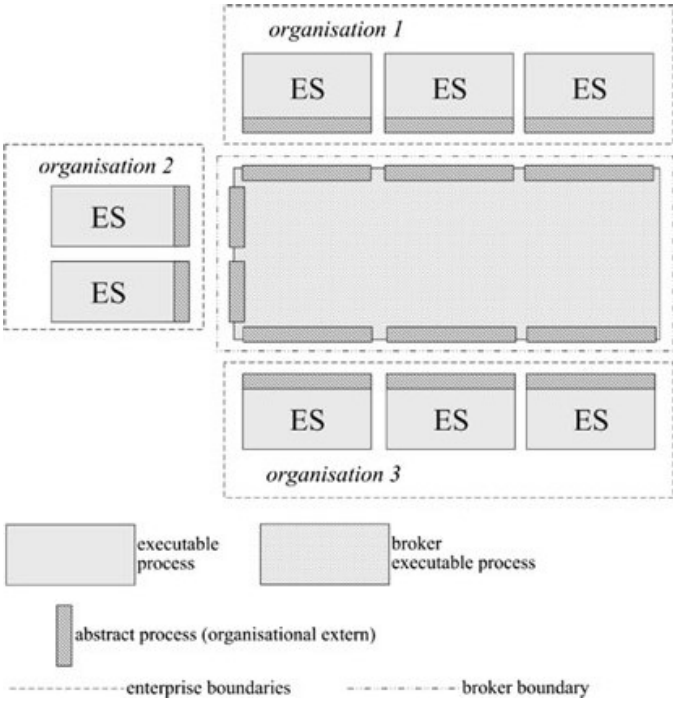


Fig. 3 Central broker architecture

The evaluation and decision method is built on integrating three approaches:

- Scenario descriptions from the *Architecture Tradeoff Analysis Method (ATAM)* (Bass et al. 2003; Bennett 1997) are used to model qualitative, architectural aspects of CBP solutions.
- *Analytic Hierarchy Process (AHP)* (Saaty 1980) is applied to conduct relative measurement and to deal with complexity and subjectivity of decisions and ratings.
- The *Multi-criteria Decision Model* by Ghand-foroush et al. (1985) is used to balance qualitative and quantitative aspects via e.g. sensitivity analysis.

Figure 5 depicts the design of our multi-criteria decision model developed to evaluate ICT architectures for CBP enactment. Its basic structure is based on the

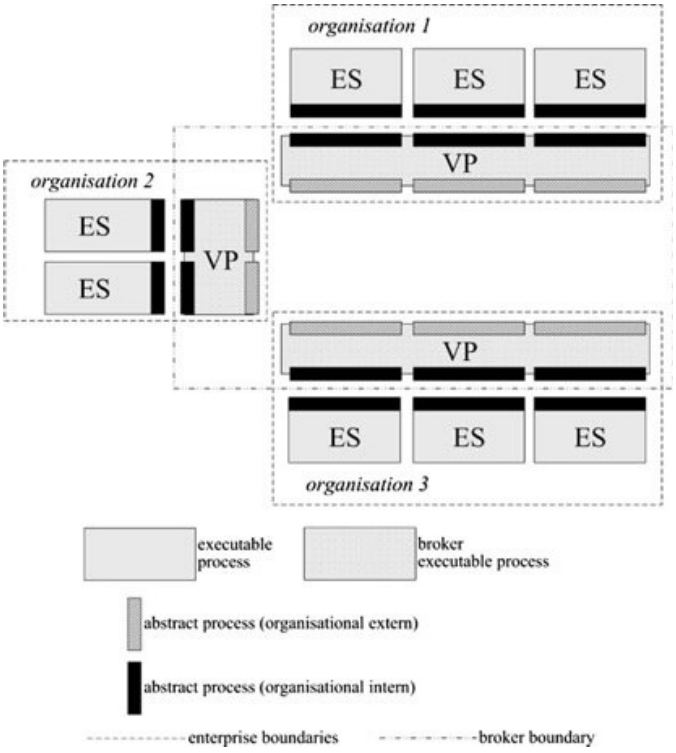


Fig. 4 Decentralized broker architecture

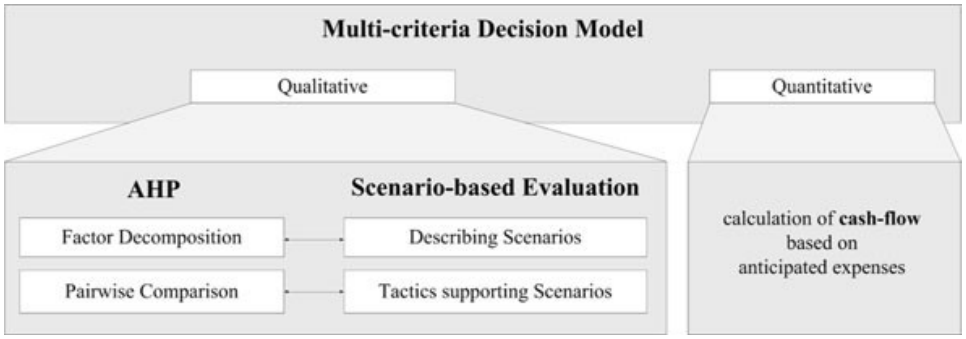


Fig. 5 Multi-criteria decision model for ICT architectures

multi-criteria decision model of Ghand-foroush et al. (1985), which is a modified version of Brown and Gibson's model (1972). It distinguishes quantitative measures from qualitative factors, and provides means for the comparison of different alternatives. This model lends itself well to conduct sensitivity analyses for architectural alternatives (by different weight of qualitative and quantitative aspects). For such rather high-level IT architecture and coordination patterns this kind of sensitivity analysis has turned out to be a good method for comparing alternatives considering both qualitative and quantitative aspects.

The quantitative aspect is determined by cash-flow analysis (net present value calculation) of the predicted costs to implement and maintain CBP solutions based on the architectural alternatives. In order to obtain a quantitative measure from the qualitative factors we combine AHP and ATAM scenario-based evaluation techniques. The architectural aspects (factors), which have to be considered in the evaluation, are described by the means of quality attributes and scenarios. They

allow to rate the evaluated architectures in terms of how well they support the respective scenarios via architectural tactics.

Rating the different architectural alternatives is a highly complex task due to various influences that have to be considered (see also Sect. 4.3). Determining concrete values for how well an alternative supports architectural qualities often involves subjectivity in the ratings to a certain degree. To deal with these challenges we apply AHP techniques to rate the alternatives. The evaluation of (overall) quality is decomposed into concrete architectural quality attributes and scenario descriptions, which are arranged in a hierarchical decomposition tree. Rating the quality attributes and the scenarios is done by pairwise comparison. The ratings, i.e. the pairwise comparisons, are based on how well the alternatives realize tactics supporting the respective scenarios. This decomposition reduces the complexity of each decision. The pairwise comparisons make it possible to rate the alternatives, without having to determine concrete values for how well an alternative supports architectural qualities.

4.1 Evaluation methodology

Evaluations conducted with the proposed decision method follow the procedure depicted in Fig. 6. In Phase (a) the decision model for the specific decision problem are set up, e.g. for CBP architecture evaluation. The selection (Step 1 and Step 2) of the quantitative and qualitative factors is optional, since the predefined factors for CBP architecture evaluation described in this article can be used. In Step 3 the prioritization of the subjective factors is determined by pairwise comparisons; this step is specific to the collaboration and organization(s) for which the evaluation is performed. Phase (b) comprises the rating the possible alternatives. In Step 1 the alternatives that shall be evaluated are chosen. The quantitative factors are rated in terms of money and person months in Step 2. The qualitative factor ratings are determined by pairwise comparison of the alternatives (Step 3). Step 4 aggregates the qualitative factor ratings to obtain the qualitative factor measure. A sensitivity analysis is performed in Phase (c) by varying the importance of the quantitative and the qualitative factor measures. Finally, one alternative is chosen in the last Phase (d).

4.2 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]

[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 software has to be developed independent of the chosen architecture.

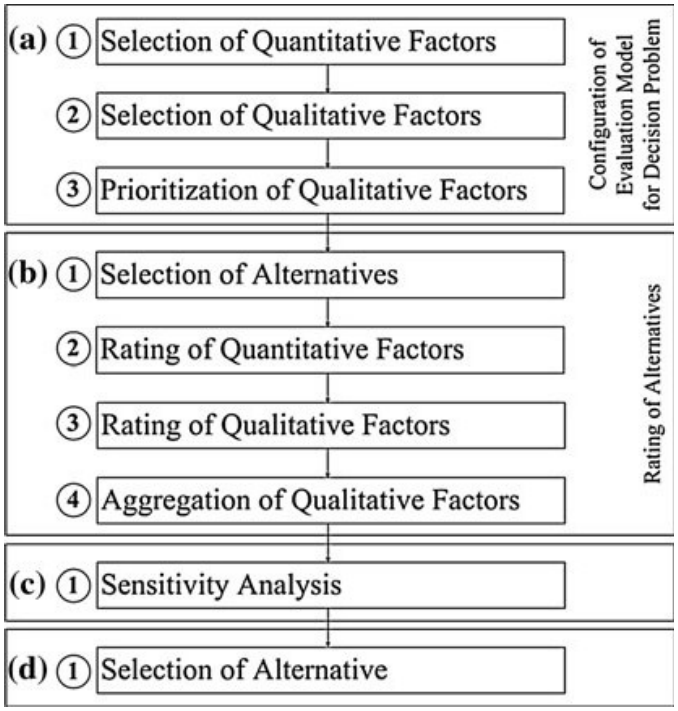


Fig. 6 Methodology for evaluation and decision model

– *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 measured in qualitative terms.

The principle underlying the model is to combine the two evaluation factors into a common evaluation measure. This requires quantitative as well as qualitative considerations, where the latter have to be transformed in common measurable units. The model allows us to select one software architecture pattern from a given set of alternatives. Following (Ghandforoush et al. 1985), 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, $1 \leq i \leq n$.

AEM_i is a measure between 0 and 1 for a particular software architecture pattern, where software architecture patterns with a higher score for the measure are considered better than patterns with a lower score. The measure depends to a large extent 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 an index with the dimension of 0:

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

where OFC_i total objective factor costs for software architecture pattern i.

Brown and Gibson (1972) employ three principles in order to ensure that the objective factor measure is compatible with the subjective factor measure: (1) the software architecture pattern with the highest cost will have the minimum OFM_i; (2) the relationship of OFC_i for each pattern relative to all other patterns is preserved; and (3) the sum of all OFM_i is equal to 1. 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, o_jm total number of first-level factors among the subjective factors total number of second-level factors in a specific first-level factor j.

The subjective factors can be grouped into a hierarchy of factors. A first-level factor 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 factor 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 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 1. Therefore, also the subjective factor measure SFM_i sums up to 1 and is represented in the same numerical scale as the objective factors.

4.3 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 proven way to determine quality attributes of software architecture. The AHP 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 methods, first-level factors are represented by quality attributes and second-level factors are represented by scenario descriptions.

Since providing sensible scales for measuring the response value of our high level software architectural patterns is difficult, we make use of pairwise comparison (see Sect. 2.3.2) to rate the qualitative factors and the evaluated software architecture patterns. The decisions for the comparisons are made based on which tactics the evaluated software architecture patterns support and on the contingency factors influencing organizations and their collaboration.

4.3.1 Scenario-based ICT architecture evaluation

Scenario-based ICT architecture evaluation (cp. Sect. 2.3) 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 well quality attributes are supported by software architecture pattern. Quality attributes and scenarios descriptions are used to determine the qualitative factor measure.

4.3.1.1 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 patterns. We do not consider execution-related topics like intrusion, or denial-of-service attacks. In the case of interoperability, which can be observed both at design and execution 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.

4.3.1.2 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 well 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 by using the schema for describing scenarios as introduced in Sect. 2.3. We refer to (Roser 2008) for the complete description of all scenarios.

Table 1 Scenario 1—modification of CBPs

Source	Management
Stimulus	Due to the constant and rapid change in business, existing CBPs have to be adapted to 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	*Brokerless:* up to n ESs of the partners are affected *Central broker:* the central broker is affected *Decentralized broker:* VPs of the respective partner(s) are affected

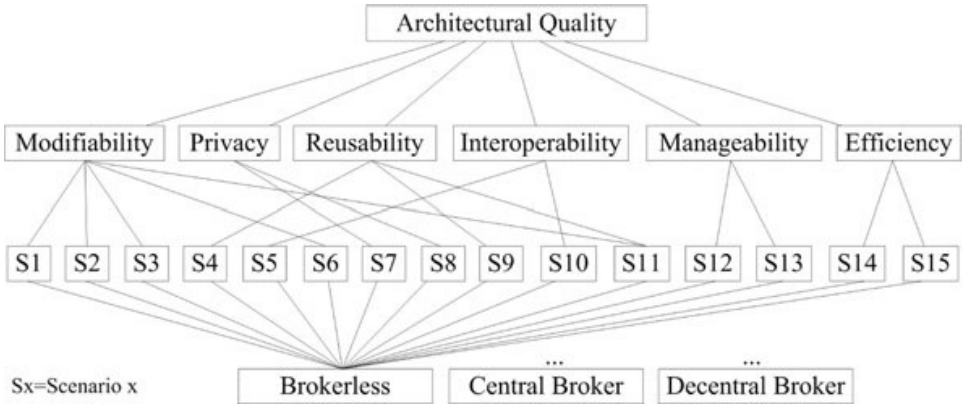


Fig. 7 AHP decomposition tree for CBP evaluation model

4.3.2 Factor decomposition and pairwise comparisons

Factor decomposition and pairwise comparisons of our evaluation model are based on the AHP.

4.3.2.1 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 Fig. 7): At the top level one can find the overall goal to have the best *architecture quality*. 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.

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

4.3.2.3 Weighting the subjective factors To determine the weights for subjective factors, i.e. which scenario 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 1 to 9 (1 means that both factors are equally important; 9 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.

$$v_i = \frac{\sum_{j=1}^{n} \overline{a_{ij}}}{n} = \frac{\sum_{j=1}^{n} \overline{\frac{a_{ij}}{\sum_{i=1}^{n} a_{ij}}}}{n} \tag{7}$$

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.

4.3.2.4 Rating the scenarios To rate the scenarios, our decision method applies a relative measurement, which is 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 over 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 (Saaty 1994, 1999). 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 well CBP coordination architectures support the scenarios are 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.

4.3.3 Rating the ICT architecture alternatives

Comparing the ICT coordination architectures requires to know how well 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 (Bass et al. 2003), software architects use so-called tactics to achieve quality attributes (see Sect. 2.3).

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. The tactic *anticipate expected changes* reduces the number of 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 *decentralized 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*. In contrast, the *brokerless architecture* realizes none of these patterns and tactics for the artifact CBPs of scenario 1. Thus we can infer that the decentralized broker architecture supports scenario 1 better than the brokerless architecture. Now, how do contingencies influence the ratings and the distance between the ratings of the evaluated architectures?

4.3.4 Influences on the ratings

The ratings of the architecture alternatives are influenced by various factors. In the following, we discuss for CBPs how tactics and patterns, that are used to achieve architectural quality, as well as contingencies, i.e. internal and external factors of the application scenario, influence the ratings.

4.3.4.1 Tactics and patterns To compare the software architectures in the *Rating of Qualitative Factors* step of the evaluation methodology (see Sect. 4.1), it is necessary to understand by which means an architect influences the quality attributes of an architecture. As described in Sect. 2.3, software architects use so-called tactics to achieve quality attributes.

Table 2 provides an overview about which tactics and patterns are most sensibly applied in an ICT coordination architecture for CBPs. 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. The tactic *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 or not one architecture variant supports a scenario better than another. Revisiting scenario 1 (cp. Table 1), the decentralized 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 brokerless architecture does realize none of the these patterns and tactics for the artifact (CBPs) of scenario 1. Thus, we can infer that the decentralized broker architecture is better suited to support scenario 1 than the brokerless architecture is.

4.4 Measuring quantitative factors

In the decision method quantitative factors are evaluated in monetary terms on the basis of the discounted cash flow approach.

Table 2 Patterns and tactics that can be used to support scenario 1–11

	Patterns					Tactics						
	Wrapper	Broker	Abstraction	Loose coupling	Orchestration	Maintain semantic coherence	Anticipate expected changes	Generalize module	Restrict communication paths	Use an intermediary	Maintain existing interfaces	Hide information
Scenario 1	–	–	x	x	x	x	x	x	–	–	–	–
Scenario 2	–	x	–	x	x	–	x	–	–	–	x	–
Scenario 3	–	x	x	x	x	x	x	–	x	–	–	–
Scenario 4	–	x	–	–	x	x	–	x	–	x	–	x
Scenario 5	–	x	x	–	x	–	x	–	–	x	x	x
Scenario 6	x	x	–	–	–	–	x	–	–	x	x	x
Scenario 7	x	x	x	–	–	–	–	–	–	x	x	x
Scenario 8	–	x	x	–	x	–	–	x	–	–	x	x
Scenario 9	–	x	x	–	x	–	–	x	–	x	x	–
Scenario 10	–	x	x	–	x	–	x	–	x	x	–	–
Scenario 11	–	–	x	x	x	x	x	–	–	x	–	–

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:

$$OFC_i = \sum_{j=1}^{m} FV_{i_j}^D = \sum_{t=0}^{N-1} \frac{FV_{i_{j_t}}}{(1+d)^t} \tag{8}$$

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.

For simplicity reasons we assume that all expenses necessary to set up an ICT system occur at present time ($FV_{i_0}; t = 0$). Running costs like for maintenance of the system or changes to the systems are considered annually ($FV_{i_t}; t > 0$). 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*).

4.5 Discussion

In this section, we discuss further issues related to the method developed in Sect. 4, including its applicability, consistency and subjectivity of rating, and the influence of contingencies.

4.5.1 Applicability of the method

4.5.1.1 Collecting input data Like other architecture evaluation methods (e.g. ATAM) our evaluation and decision method is quite data intensive and requires the involvement of different stakeholders in its application: IT managers, architects, as well as the customers of the project. To obtain the input from the people involved in the evaluation, the well-known *Delphi* approach (Sackmann 1974) can be used, where independent experts estimate e.g. ratings and costs. Depending on their calculations, either a workshop can be set up to discuss the results, or a coordinator normalizes the data provided by the experts. This normalized and harmonized data is the input for our evaluation and decision method.

Moreover, there exist various possibilities to collect the necessary data more easily. When applying the evaluation method in a reorganization and reenginering context, quantitative data (software, hardware, labor; cp. Sect. 4.4) is already available. Another possibility is the construction and usage of evaluation databases. Their data can be used to derive the data for qualitative measurements, in particular the factor decomposition and the pairwise comparisons, for evaluations within a similar context. The scenario descriptions for CBP-environments described in Sect. 4.3.1 normally stay the same for the various evaluations or are reused with only small modifications.

4.5.1.2 Cross-organizational vs. intra-organizational application The evaluation and decision method is very well applicable for (distributed but) intra-organizational

business processes, where the partners with quite similar metrics and standards cooperate. However, we explicitely designed the evaluation and decision method in way, that it can be also applied to cross-organisational business processes. The method is solely based on comparing the architectural alternatives, via quality attributes and scenarios we developed and described in Sect. 4.3.1, and not the collaborating partners. Because of these AHP comparisons of the architectural alternatives, no explicit scales, standards, or comparable IT figures need to be established between the collaborating partners.

4.5.1.3 Measuring architectural qualities A common approach to architecture evaluation is to describe architectural quality characteristics via quality attributes. Since these quality attributes cannot be quantified directly, we need to measure quality attributes and scenario descriptions in qualitative terms.

The Cost Benefit Analysis Method (CBAM) (Clements et al. 2002; Kazman et al. 2001), which is build on ATAM, aims to model the costs and the benefits of architectural design decisions and is a means of optimizing such decisions. The idea behind the CBAM is that architectural strategies affect the quality attributes of the system and these in turn provide system stakeholders with some benefit, called utility in this context. Each architectural strategy provides a specific utility level to the stakeholders. Each of them also has cost and takes time to implement. Given this information, the CBAM can aid the stakeholders in choosing architectural strategies based on their return on investment, i.e. the ratio of benefit to cost.

In CBAM, stakeholder benefit is expressed in so-called utility-response curves. A utility-response curve depicts how the utility derived from a particular response varies as the response varies (for response cp. the scenario description in Sect. 2.3). As an example, a concave utility-response function would value a very high availability in response to failure only slightly more than moderate availability; however, low latency might be valued substantially more than moderate latency. As described in (Clements et al. 2002), eliciting the utility characteristics from the stakeholders can be a tedious process. The authors suggest to elicit only rough approximations of these curves from the stakeholders, using five values of the quality attribute response and to interpolate the rest of the curve.

However, as Saaty argues in (1994), an appropriate scale is a prerequisite to obtain sensible measurements. When evaluating architectures of concrete software systems, it is already a difficult task to obtain appropriate scales for properties like latency or response-time (in e.g. milliseconds) in relation to their utility. For most properties (architectural qualities) relevant for our rather abstract architecture patterns for ICT system coordination, there is no standard scale of measurement. In our evaluation and decision method we therefore apply the approach of deriving relative scales and relative measurements through pairwise comparisons like those introduced by the AHP. This allows us to measure architectural qualities without the need to determine a given scale and to consider intangible properties.

4.5.1.4 Scalability Applying AHP techniques for measuring architectural quality involves a considerable number of pairwise comparisons. In Moore et al. (2003) the

authors report, that for their architecture evaluations between 300 and more than 1500 comparisons would have been necessary with AHP. However, for our method, as described in this article including quality attributes and scenario descriptions, only 77 comparisons are necessary to evaluate candidate architectures (see also the case studies in Chap. 5).

This rather big difference stems from the different kinds of alternatives that are evaluated. In Moore et al. (2003) the authors aimed to select a subset from 58 architectural strategies (cp. Chap. 2.3), i.e. they had a rather huge number of alternatives. In our evaluation and decision method, the alternatives in question are much more coarse-grained: they are whole candidate architectures. Hence, there are only 77 necessary comparisons for three candidate architectures as described in this article. For four and five candidate architectures 122 and 182 comparisons would be necessary.

The number of comparisons will increase, as our method is adjusted to other evaluation scenarios, where more quality attributes and scenario descriptions need to be considered. For such a potential evaluation scenario with 10 quality attributes and 25 scenario description for three/four/five candidate architectures, one would need about 150/230/330 comparisons, respectively. Further increasing the scalability of our approach is an area of future work: for larger scenarios, it may be sensible to introduce a further hierarchy level in the decomposition tree. This also reduces the number of comparisons, since less factors in level x of the decomposition tree need to be compared to obtain the measure for a factor in level x-1 of the decomposition tree.

4.5.2 Consistency and subjectivity of ratings

In ranking processes there is often a large amount of subjectivity through the people that conduct ratings (cp. lessons learned in Moore et al. 2003). One can deal with this problem like in other evaluation methods, e.g. by conducting a workshop led by independent consultants and inviting people with different backgrounds. The goal is to obtain better results through discussion and exchange of experiences by those people.

However, the AHP method provides another mechanism to identify ratings that might not be correct and need further discussion. The AHP method involves redundant comparisons to improve validity, recognizing that participants may be uncertain or make poor judgements in some of the comparisons. This redundancy leads to multiple comparisons that may lead to numerical inconsistencies. A consistency ratio (CR) is used to estimate the accuracy of the data. The consistency ratio can be calculated on the basis of the maximum eigenvalue λ_{max} of the comparison matrix and a coefficient RI[5]: $CR = CI/RI = \frac{\lambda_{max}-n}{n-1}/RI$. Saaty suggests that errors in the measurements are tolerable when $CR \leq 10\%$ (Saaty 1999).

[5] The author of (Saaty 1999) suggests to use the following values for RI: $n = 3$, $RI = 0.52$; for $n = 4$, $RI = 0.89$; for $n = 5$, $RI = 1.11$; for $n = 6$, $RI = 1.25$.

For example, if alternative a_1 is better than alternative a_2 by a factor 3, and alternative a_2 is better than alternative a_3 again by a factor 3, than alternative a_1 has to be better than alternative a_3. For example, the factor 3 for the pairwise comparison between a_1 and a_3 would not be sufficient, since this would imply that a_1 is equal to a_2 ($CR = 13\%$).

4.5.3 Influence of contingencies

The ratings of architectures depend to a high degree on the organizational, the collaboration, and the external context for which the architectures are evaluated. Like in contingency theory (Donaldson 2001) (see Sect. 2.3) where performance and efficiency of an organizational structure depend on internal and external contingencies, the rating of the evaluated architectures depends on such contingencies.

Contingencies influence the scenario ratings. When comparing two alternatives and determining how big the difference between them is, we always have to take these contingencies into account. In the following, we provide an overview of the contingencies relevant for our decision method and show how they can be used to make the pairwise comparisons for the scenarios easier.

For our decision method it is necessary to 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 the distribution of influence and power among the partners into account; 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 the *number of process steps*, defined through the number of processing steps, or the estimated number of process instances during execution. External contingencies are external factors that highly influence organizations' decisions and strategies, and that therefore also impact the choice of ICT coordination architectures: *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 (Legner and Wende 2006).

Table 3 illustrates how contingencies influence the scenario ratings. Depending on the contingencies the importance of realizing tactics to best support scenarios varies. In the case this is directly proportional ($\uparrow\uparrow$), we can say that the stronger the influence of contingencies (e.g. higher product complexity) is, the higher is the difference between two architecture ratings, where one architecture supports and the other architecture does not support the scenario by tactics. Inversely proportional ($\uparrow\downarrow$) represents the fact that a higher influence of contingencies (e.g. high degree of standardization) results in a lower difference between the architectures in the scenario ratings.

Table 3 Influence of contingencies on scenario ratings

		Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5
Internal contingencies	Coll. topology	Initiator ⇑	Power of players ⇑	n/a	n/a	n/a
	Product	Complexity ⇑	Complexity ⇑	n/a	Complexity/ specifity ↑↓	n/a
	Service flow	n/a	n/a	n/a	n/a	⇑
	Process	n/a	# Process steps ⇑	# Process instances ⇑	# Process steps ⇑	# Process steps ⇑
External conting.	Standardization	↑↓	↑↓	↑↓	↑↓	↑↓
	Maturity	↑↓	↑↓	↑↓	↑↓	↑↓
	Bus. semantics	n/a	↑↓	↑↓	↑↓	↑↓
	Legislation	n/a	n/a	n/a	n/a	n/a

If we assume for example a high degree of standardization to rate scenario 1, the decentralized broker architecture is not much better than or even equal to the brokerless architecture. 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.

Other contingencies exist, 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 weighting scenarios and quality attributes. Change is not addressed a second time in rating the scenarios.

5 Applying the evaluation method

The evaluation method described in Sect. 4 has been developed based on experiences made with the development of different enterprise interoperability use cases, which were modelled and prototypically implemented as part of the European Integrated project ATHENA.[6] In ATHENA four application scenarios were considered: (1) an Automotive virtual enterprise use case for strategic sourcing (one OEM, multiple tier-one suppliers); (2)an aerospace collaborative engineering use case (one aircraft manufacturer, many small engineering service suppliers); (3) a use case involving collaboration between second-tier and third-tier suppliers, mostly small or medium enterprises (SMEs) in the automotive domain; (4) a furniture supply chain requiring collaboration between SMEs. These scenarios were analyzed, model-driven engineering solution including corresponding models and model transformations were devised, and IT architectures were developed (see e.g.

[6] FP6-IST-507849, information available at http://cordis.europa.eu/fp6/projects.htm

(Stäber et al. 2007; Stäber and Müller 2007)). The method as described in this paper bundles lessons learnt, experiences, and best practices gathered during this project. In this section, we apply the evaluation method to two of these use cases: the virtual enterprise automotive use case (1) (Sect. 5.1) and the automotive SME use case (3) (Sect. 5.2). Within this section, we present the use cases in a way that abstracts from the specific vertical domain but retains the general principles of CBP modeling and enactment. 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 costs (quantitative factors) through the choice of a collaboration architecture is discussed in a sensitivity analysis.

5.1 Virtual enterprise use case

This use case deals with virtual enterprises that collaborate in big, long-running CBPs (approx. 90 processing steps). An OEM and big first-tier suppliers 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 each other's markets. About half of the services on the supplier side, that are visible in the CBP are 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. In the next paragraphs we present the results of applying our evaluation method to the virtual enterprise use case. The complete data relevant for the evaluation can be found in (Roser 2008).

5.1.1 Configuration of the decision model

For this evaluation we measure the quantitative factors in terms of *software*, *labor*, and *hardware* as described in Sect. 4.4. The quality attributes and scenarios described in 4.3.1 are used as qualitative factors.

5.1.1.1 Prioritization of qualitative factors To determine the weight of the quality attributes and the scenarios, pairwise comparison is applied as described in Sect. 4.3.2.

Table 4 depicts the weighting of the first-level factors, i.e. the quality attributes, for the virtual enterprise use case. 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.

Table 5 depicts the weighting of the scenarios which are used to describe the modifiability attribute in the virtual enterprise use case. The scenarios are compared in the same manner as the other quality attributes in Table 4. The column of the priority vector v_i depicts the weighting of the scenarios.

Table 4 Priority comparison matrix for the first-level factors

	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
Manageability	$\frac{1}{3}$	5	1	$\frac{1}{5}$	1	1	0.10

Table 5 Priority comparison matrix for the second-level factor modifiability

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

5.1.2 Rating of the alternatives

The three architecture patterns *brokerless*, *central broker*, and *decentralized broker architecture* introduced as in Sect. 3 are considered as alternatives for the evaluation.

5.1.2.1 Rating of quantitative factors The objective measure is calculated on the basis of the cash flow of the costs for software, labor, and hardware. For the virtual enterprise use case 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 brokerless architecture 5075 thousand cost units were estimated (OFC_i). The overall objective measure OFM_i can be found in the rightmost column.

5.1.2.2 Rating of qualitative factors The scenarios are rated by pairwise comparison of the architecture alternatives. The decisions are based on how well 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 use case for which the evaluation is performed.

Table 7 depicts the rating matrix for scenario 1. As described in Sect. 4.3.3, the central broker (CBr) alternative supports scenario 1 better than the brokerless (BrL) alternative. Relevant contingencies for scenario 1 are the grade of standardization and the maturity of the CBP and the services. Since both contingencies are rather

Table 6 Overall objective measure

	Software (K)	Hardware	Labour (K)	OFC_i (K)	OFM_i (K)
Brokerless (BL)	45	75	4,955	5,075	0.127
Central Broker (CBr)	69	95	1,200	1,364	0.474
Decentralized Broker (DBr)	118	135	1,367	1,620	0.399

Table 7 Rating scenario 1

Scenario 1	BrL	CBr	DBr	v_i^{id}
Brokerless (BrL)	1	$\frac{1}{7}$	$\frac{1}{7}$	0.14
Central Broker (CBr)	7	1	1	1.00
Decentralized Broker (DBr)	7	1	1	1.00

low in the virtual enterprise use case, the architectural quality is important for the support of this scenario, which leads to a comparison value of 7 between the central broker and brokerless architecture. The central and decentralized broker (DBr) architecture a rated equally important, i.e. the value is 1. The column of the priority vector v_i^{id} depicts the weighting of the scenarios, as introduced in Sect. 4.3.2.

5.1.2.3 Overall subjective measure The overall subjective measure is computed on the basis of the factor weights and the scenario ratings. Table 8 depicts the relevant data. Row 2 contains the weighting of the quality attributes from Table 4. 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 by applying formula (3) and can be found in the rightmost column.

5.1.3 Sensitivity analysis and selection of alternative

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 can also be used for sensitivity analysis.

Figure 8 depicts the sensitivity analysis chart for the virtual enterprise use case. The x-axis represents the importance of the objective factors measure; the y-axis gives 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 decentralized broker architecture variant should be selected. The brokerless variant gets significantly lower rating values for all X than the other ones. The decentralized broker architecture scores better for the qualitative measurements (especially for $X = 0$), while the central broker architecture is better in terms of IT

Table 8 Overall subjective measure (BrL = Brokerless, CBr = Central Broker, DBr = Decentralized Broker)

	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	0.25	0.75	0.67	0.33		
BrL	0.14	0.11	0.08	0.16	0.08	0.24	0.20	0.25	0.17	0.08	0.14	0.17	1.00	0.17	0.12	0.11	0.178	0.108
CBr	1.00	0.44	0.30	0.46	0.30	0.14	0.20	0.25	0.59	0.30	1.00	0.41	0.33	1.00	1.00	1.00	0.568	0.343
DBr	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	0.908	0.549

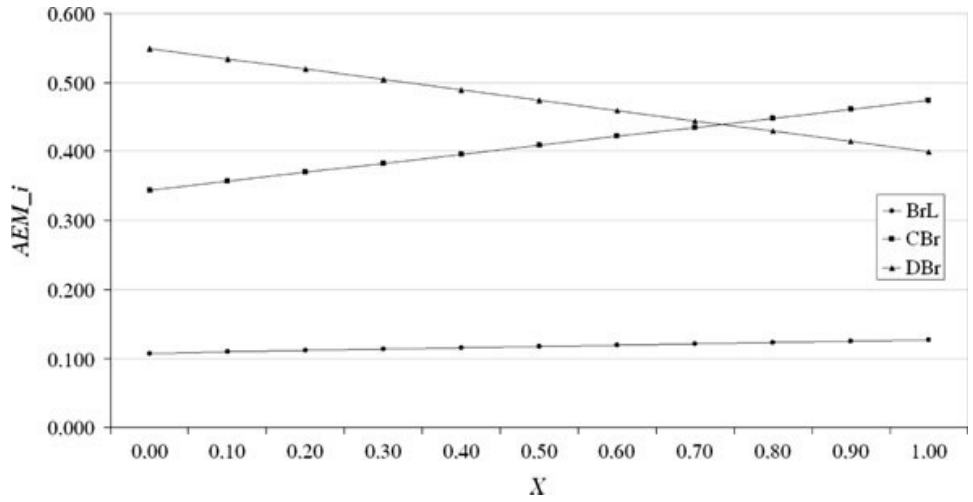


Fig. 8 Sensitivity analysis chart (*BrL* Brokerless, *CBr* Central Broker, *DBr* Decentralized Broker)

Table 9 Priority comparison matrix for the first-level factors

	Mod.	Pri.	Reuse	Int.	Eff.	Man.	v_i
Modifiability	1	4	$\frac{1}{2}$	3	$\frac{1}{2}$	$\frac{1}{2}$	0.16
Privacy	$\frac{1}{4}$	1	$\frac{1}{5}$	$\frac{1}{2}$	$\frac{1}{5}$	$\frac{1}{5}$	0.04
Reuse	2	5	1	4	1	1	0.25
Interoperability	$\frac{1}{3}$	2	$\frac{1}{4}$	1	$\frac{1}{4}$	$\frac{1}{4}$	0.06
Efficiency	2	5	1	4	1	1	0.25
Managability	2	5	1	4	1	1	0.25

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. Strassmann 2006). This leads to an estimation of $X \approx 0.2$ for the virtual enterprise use case applied to the automotive industry. Thus, we would suggest to select the decentralized broker architecture in the virtual enterprise use case. Even if transaction costs and IT costs were equally important ($X = 0.5$), the architecture evaluation measure of the decentralized broker variant would be still be slightly preferable to the central broker variant.

5.2 SME use case

This use case represents the CBPs between the second-tier (or even third- and fourth-tier suppliers) in the automotive sector. 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 eBusiness eXtensible Markup Language (ebXML) Business Process Specification Schema (BPSS) (OASIS 2006)). Hence, interoperability is not such an important issue for these organizations. Also, changes to the existing CBPs are rare (up to three times a year). However, about 50% of the services visible on the SME side are based on legacy applications, which will be partially replaced within the next five years. In the next paragraphs we will present the results of applying our evaluation method to the SME use case. The complete data relevant for the evaluation can be found in (Roser 2008).

5.2.1 *Configuration of the decision model*

For this evaluation we measure the quantitative factors in terms of *software*, *labor*, and *hardware* as described in Sect. 4.4. The quality attributes and scenarios described in 4.3.1 are used as qualitative factors.

5.2.1.1 Prioritization of qualitative factors To determine the weight of the quality attributes and the scenarios, pairwise comparison is applied as described in Sect. 4.3.2.

Table 9 depicts the weighting of the first-level factors, i.e. the quality attributes, for the SME use case. 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.

Table 10 depicts the weighting of the scenarios which are used to describe the modifiability attribute in the SME use case. The scenarios are compared in the same way as the other quality attributes in Table 9. The column of the priority vector v_i depicts the weighting of the scenarios.

5.2.2 *Rating of the alternatives*

As alternatives for the evaluation, we consider the three architecture patterns *brokerless*, *central broker*, and *decentralized broker architecture* as introduced in Sect. 3.

Table 10 Priority comparison matrix for second-level factor modifiability

	Sc. 1	Sc. 2	Sc. 3	Sc. 6	Sc. 11	v_i
Scenario 1	1	$\frac{1}{9}$	1	$\frac{1}{5}$	$\frac{1}{3}$	0.05
Scenario 2	9	1	9	5	7	0.59
Scenario 3	1	$\frac{1}{9}$	1	$\frac{1}{5}$	$\frac{1}{3}$	0.05
Scenario 6	5	$\frac{1}{5}$	5	1	3	0.21
Scenario 11	3	$\frac{1}{7}$	3	$\frac{1}{3}$	1	0.11

Table 11 Overall objective measure

	Software (K)	Hardware (K)	Labour (K)	OFC_i (K)	OFM_i (K)
Brokerless (BrL)	100	100	100	300	0.453
Central Broker (CBr)	124	120	195	439	0.310
Decentralized Broker (DBr)	197	180	198	575	0.237

Table 12 Rating scenario 1

Scenario 1	BrL	CBr	DBr	v_i^{id}
Brokerless (BrL)	1	$\frac{1}{2}$	$\frac{1}{2}$	0.50
Central Broker (CBr)	2	1	1	1.00
Decentralized Broker (DBr)	2	1	1	1.00

5.2.2.1 Rating of quantitative factors The objective measure is calculated on the basis of the cash flow of the costs for software, labor, and hardware. For the SME use case with four collaborating enterprises, we have estimated the following costs, see Table 11. 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. Table 11 reveals that for the brokerless architecture 300 thousand cost units were estimated (OFC_i). The overall objective measure OFM_i is shown in the rightmost column.

5.2.2.2 Rating of qualitative factors The scenarios are rated by pairwisely comparing the architecture alternatives. The decisions are based on how well 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 use case for which the evaluation is performed.

Table 12 illustrates the rating matrix for scenario 1. The column of the priority vector v_i^{id} depicts the weighting of the scenarios.

5.2.2.3 Overall subjective measure The overall subjective measure is computed on the basis of the factor weights and the scenario ratings. Table 13 shows the relevant data. Row 2 illustrates the weighting of the quality attributes from Table 9. 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 ratings, 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 by using the formula (3) and can be found in the rightmost column.

5.2.3 Sensitivity analysis and selection of alternative

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

Table 13 Overall subjective measure (*BrL* Brokerless, *CBr* Central Broker, *DBr* Decentralized Broker)

	Modifiability					Sec.		Reuse			Int.op.		Eff.		Man.		SFM_i^{id}	SFM_i
	0.16					0.04		0.25			0.06		0.25		0.25			
	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		
BrL	0.50	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	0.530	0.293
CBr	1.00	0.50	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	1.00	0.589	0.326
DBr	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	0.55	0.41	0.688	0.381

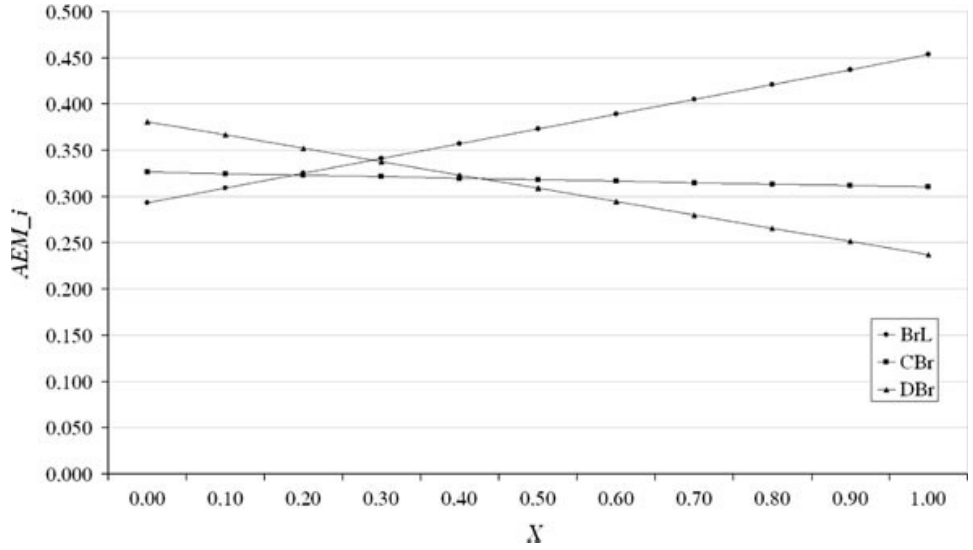


Fig. 9 Sensitivity analysis chart (*BrL* Brokerless, *CBr* Central Broker, *DBr* Decentralized Broker)

factor measure SFM_i (see formula (1)). The measure depends on the weight X assigned to the objective and subjective factor. This weight can also be used to perform a sensitivity analysis for the SME use case, as illustrated in Fig. 9. The x-axis represents the importance of the objective factors measure, while the y-axis shows the architecture evaluation measure for the respective architecture variant.

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 brokerless architecture variant 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, e.g. caused by new governmental monitoring requirements, 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 SME use case.

6 Conclusion

As the scope of EAI reaches beyond the boundaries of individual enterprises, ICT systems need to be able to support the flexible design and automation of cross-organizational business process coordination. Therefore, new constructs, models and methods are required to support enterprise modelers and ICT architects in developing ICT solutions tailored to business requirements. Yet, as we found out when modeling a diverse sample of application scenarios in the ATHENA EU project (see Sect. 5), as per today, these constructs, models, and methods are not available in a form that is ready to use.

The objective of the work described in this paper is to take a first step towards providing ICT architects with an evaluation and decision model that enables the

principled assessment and selection of "the right" ICT architecture for a given CBP use case. Following the principles of design science, the contribution of this paper has been threefold. First, we presented three common architectural patterns for (service-oriented) CBP coordination. Second, the core contribution is established by a new method for decision support suitable for ICT architects to derive and evaluate an appropriate architecture paradigm for a given use case or application domain. Third, the method is accompanied by a set of representative scenario descriptions that support the evaluation and selection of appropriate ICT system coordination architecture paradigms for CBP enactment, as well as a set of guidelines for how different contingencies influence ICT system coordination architecture.

The evaluation and decision support method can be used to validate guidelines like "Standardization is very important for the brokerless approach …", "The hierarchical structure is typically used in traditional business relationships where interactions between cooperating partners are agreed to in advance.", or "Process management and monitoring of the overall process status is generally easier in hierarchical structures …" described in [p. 204f, 21]. We further use the method to develop new guidelines for ICT architecture selection based on the influence of contingencies.

The evaluation enables enterprise ICT architects to develop a better understanding of the influence of contingencies on the overall decisions and to apply these guidelines. Our experience so far indicates that pairwise comparisons reduce the amount of information that is necessary for decisions. Humans can only deal with information that involves a small number of facts simultaneously; therefore, 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 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.

In summary, our work builds on experience and lessons learned from the ATHENA use cases in order to close a gap and to differentiate itself from other research on architecture evaluation (see Sect. 2) by focusing on the decision-making processes involved in the development of ICT solutions for CBP coordination and automation.

One limitation of our approach includes scalability issues resulting from the use of AHP (see discussion above and in Sect. 4.5.1). As indicated in Sect. 4.5.1), hierarchical decomposition is a promising way of tackling these issues; investigating how and to what extent scalability of the method can be improved is an area of future work. Another open field is to explore how decision methods like the one presented in this paper can be built into existing enterprise modeling frameworks and model-driven IDEs, to support process modelers and ICT architects in their tasks of creating and managing executable CBP specifications from business-level models. A third area concerns the specification of further scenarios to extend the scope of applying the evaluation and decision method. More fine-grained models and extensions of our decision method need to be developed to support the process down to the platform-specific and code levels. Also, while the method has been

informed by extensive experience with service-oriented CBP applications gained during the ATHENA project, and is to our knowledge the first evaluation and decision-making method available geared to the specific requirements of service-oriented CBP deployment scenarios, more empirical research is required to validate the approach for its usability as well as its usefulness for domains that differ from the ones investigated in so far (automotive, aerospace, furniture supply chain). future proof via practical validation.

References

Arkin A (2002) Business process modeling language (BPML). BPMI.org

Bass L, Clements P, Kazman R (2003) Software architecture in practice. Addison-Wesley, Massachusetts

Bass L, John BE (2003) Linking usability to software architecture patterns through general scenarios. J Syst Softw 66(3):187–197

Bennett DW (1997) Designing hard software—the essential task. Prentice Hall, NY

Birman A, Ritsko J (2005) Preface to service-oriented architecture. IBM Syst J 44(4):651–652

Brown PA, Gibson DF (1972) A quantified model for facility site selection application to multiplant location problem. AIIE Trans: Ind Eng Res Dev 4(1):1–10

Clements P, Kazman R, Klein M (2002) Evaluating software architecture. Addison-Wesley, Massachusetts

Dolan TJ (2001) Architecture assessment of information-system families. Ph.D. thesis, Technische Universiteit Eindhoven

Donaldson L (2001) The contingency theory of organizations. SAGE Publications, Inc, Los Angeles

Erl T (2004) Service-oriented architecture: a field guide to integrating XML and web services. Prentice Hall International, NJ

Erl T (2005) Service-oriented architecture: concepts, technology, and design. Prentice Hall International, NJ

Gamma E, Helm R, Johnson RE (1995) Design patterns. Elements of reusable object-oriented software. Addison-Wesley Longman, Massachusetts

Ghandforoush P, Huang PY, Taylor BW (1985) A multi-criteria decision model for the selection of a computerized manufacturing control system. Int J Prod Res 23(1):117–128

Hammer M, Champy J (1993) Reengineering the corporation: a manifesto for business revolution. Harper Business, New York

Hevner AR, March ST, Park J, Ram S (2004) Design science in information systems research. MIS Q 28(1):75–105

IBM, Systems B, Microsoft AG, S, Systems S (2003) Business process execution language for web services version 1.1

Kazman R, Asundi J, Klein M (2001) Quantifying the costs and benefits of architectural decisions. In: 23rd international conference on software engineering, pp 297–306. doi:10.1109/ICSE.2001.919103

Klein R, Kupsch F, Scheer AW (2004) Modellierung inter-organisationaler Prozesse mit Ereignisgesteuerten Prozessketten. In: Veröffentlichungen des Instituts für Wirtschaftsinformatik, vol 178, Scheer

Legner C, Wende K (2006) Towards an excellence framework for business interoperability. In: 19th Bled eConference “eValues”, Slovenia

Leymann F, Roller D, Schmidt MT (2002) Web services and business process management. IBM Syst J 41(2):198–211. doi:10.1147/sj.412.0198

Lippe S, Greiner U, Barros A (2005) A survey on state of the art to facilitate modelling of cross-organisational business processes. In: 2nd GI workshop XML4BPM at 11th GI conference BTW 2005, Karlsruhe, Germany, pp 7–22

Liu DR, Shen M (2001) Modeling workflows with a process-view approach. In: 7th international conference on database systems for advanced applications, Hong Kong, China. IEEE Computer Society, pp 260–267

Malone TW (1987) Modeling coordination in organizations and markets. Manage Sci 33(10):1317–1332

Moore M, Kazman R, Klein M, Asundi J (2003) Quantifying the value of architecture design decisions: lessons from the field. In: 25th international conference on software engineering, pp 557–562. doi: 10.1109/ICSE.2003.1201237
OASIS (2006) ebXML business process specification schema technical specification v2.0.4. ebxmlbp-v2.0.4-Spec-os-en
OASIS (2007) Web services business process execution language version 2.0. wsbpel-primer
OMG (2006) Business process definition metamodel (BPDM), final submission. bmi/2006-11-03
OMG (2006) Business process modeling notation specification, final adopted specification. dtc/06-02-01
OMG (2007) Unified modeling language: superstructure, version 2.1.1. formal/07-02-05
Roser S (2008) Designing and enacting cross-organisational business processes: a model-driven, ontology-based approach. PhD thesis, University of Augsburg. urn:nbn:de:bvb:384-opus-8057
Saaty TL (1980) The analytic hierarchy process. McGraw-Hill, New York
Saaty TL (1994) How to make a decision: the analytic hierarchy process. Interfaces 24(6):19–43
Saaty TL (1999) Decision making for leaders, 3rd edn. RWS Publications, USA
Sackmann H (1974) Delphi assessment: expert opinion, forecasting, and group process. Tech. Rep. R-1283-PR, Rand, Santa Monica. http://www.rand.org/pubs/reports/2006/R1283.pdf
Saliu O, Ruhe G (2005) Software release planning for evolving systems. Innov Syst Softw Eng 1(2):189–204. doi:10.1007/s11334-005-0012-2
Schulz KA, Orlowska ME (2004) Facilitating cross-organisational workflows with a workflow view approach. Data Knowl Eng 51(1):109–147. doi:10.1016/j.datak.2004.03.008
Snow CC, Miles RE, Coleman HJ (1992) Managing 21st century network organizations. Organ Dyn 20(3):5–20. doi:10.1016/0090-2616(92)90021-E
Stäber F, Müller JP (2007) Evaluating peer-to-peer for loosely coupled business collaboration: a case study. In: Alonso G, Dadam P, Rosemann M (eds) 5th international conference on business process management, Lecture Notes in Computer Science, vol 4714. Springer, pp 141–148
Stäber F, Sobrito G, Müller JP, Friese T (2007) Interoperability challenges and solutions in automotive collaborative product development. In: RG et al (eds) Enterprise interoperability II: 3rd international conference on interoperability of enterprise software and applications, Springer, pp 709–720
Strassmann PA (2006) Is outsourcing profitable? Lecture at George Mason University. http://www.strassmann.com/pubs/gmu/OutsourcingV6.pdf
van der Aalst WM, ter Hofstede AH (2005) YAWL: yet another workflow language. Inf Syst 30(4):245–275. doi:10.1016/j.is.2004.02.002
W3C (2002) Web service choreography interface (WSCI) 1.0. http://www.w3.org/TR/wsci
Wallis JJ, Douglas CN (1986) Measuring the transactions sector in the American economy, series, volume, Long-term factors in American economic growth edn. University of Chicago Press, Chicago
WfMC (2005) Process definition interface—XML process definition language. WFMC-TC-1025
Williamson OE (1975) Markets and hierarchies: analysis and antitrust implications. Free Press, New York
Williamson OE (1989) Transaction cost economics. Handbook Indust Organi 1:135–182. doi: 10.1016/S1573-448X(89)01006-X