# Choreography Modelling in Embedded Systems Domain
## *Requirements and Implementation Technologies*

Nebojša Taušan, Jouni Markkula, Pasi Kuvaja and Markku Oivo
*Department of Information Processing Science, University of Oulu, POBox 3000, Oulu, Finland*

Keywords: Choreography, Design Requirements, Design Science, Modelling Language, BPMN.

Abstract: Software companies that develop embedded systems following the principles of service-oriented architecture can anticipate various benefits from choreography modelling. Current choreography modelling languages, however, have a limited applicability in embedded systems development since they are not expressive enough to capture all the choreography-relevant aspects that are typical in this domain. We tackled this problem by analysing the needs of embedded systems for choreography modelling language. Our analysis was guided by design science and relied on expert interviews, company-specific documents and the relevant literature. The main results of the analysis presented in this paper are a) design requirements addressing the limitations of choreography modelling languages for embedded systems development and b) proposals for modelling language implementation technologies. The feasibility of these results is evaluated by redesigning an existing choreography modelling language and by implementing a prototype editor for the redesigned language.

## 1 INTRODUCTION

The size and the complexity of today's embedded systems (ES) have led to the growing adoption of model-driven engineering (MDE) (Liggesmeyer and Trapp, 2009) and service-oriented architecture (SOA) in their development (Cannata et al., 2008; Gilart-Iglesias et al., 2007). MDE, as an overall engineering approach, relies on practices such as modelling, model transformation and code generation to facilitate the software development practices (Schmidt, 2006). SOA relies on service interactions as a means of achieving system goals in a flexible manner (Scholz et al., 2009). A specification of service interactions can therefore be seen as an important development artefact which is commonly modelled form at least two viewpoints – choreography and orchestration (Peltz, 2003; Dijkman and Dumas, 2004).

Choreography and orchestration models capture different details relevant to the specifications of service interactions. The *choreography* captures the interactions between participants, which represent autonomous management authorities whose services are interacting to achieve a common goal (Barros et al., 2005). The *orchestration* focuses on the service interactions needed to achieve the goal of a single participant (Dijkman and Dumas, 2004). These two viewpoints can therefore be seen as complementary. Several studies, however, suggest that the interest in choreography will grow in the ES domain as the information it conveys contributes to greater scalability and performance (Kaur et al., 2013; Starke et al., 2013).

The choreography viewpoint is commonly specified using Choreography Modelling Language (CML), and once specified, it can address various misalignments in the development process. Some of these misalignments include duplication of work, delays in the development and a loss of opportunity from the parallelization of work (Taušan et al., 2014). One drawback is that currently used CMLs have limited applicability in ES development since they are not expressive enough to capture various choreography-relevant development aspects that are typical in the ES domain. This drawback is argued by presenting a review of the CMLs used in model-driven ES development alongside the development aspect they address.

Scribble language is introduced in (Hu et al., 2013) and used for the specification of a choreography protocol between concurrent software components embedded in various instruments for ocean observation. The rigorousness and verifiability of the protocol were seen as crucial aspects of these ESs. To address them, Scribble syntax was designed based on the multiparty session type theory.

Abstract Process Execution Language, or APEL, represents a hybrid modelling language that merges

Taušan, N., Markkula, J., Kuvaja, P. and Oivo, M.
Choreography Modelling in Embedded Systems Domain - Requirements and Implementation Technologies.
DOI: 10.5220/0005686700750086
In *Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2016)*, pages 75-86
ISBN: 978-989-758-168-7
Copyright © 2016 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

both choreography and orchestration viewpoints (Pedraza and Estublier, 2009). APEL specifies interactions in an orchestrated manner and uses annotations to label model parts which will be deployed to various participants. During the system execution, these parts will interact in a choreographed manner. Besides participants, APEL also allows the annotation of model parts with other ES properties.

The management of long-running transactions is seen as one of the crucial tasks in SOA-based systems (Ciancia et al., 2011). This task is partially addressed by expressing long-running transactions with choreography and orchestration models. To ensure their conformance, two models are founded on two related formal calculus – network coordination policy and signal calculus.

CHOREO is a choreography language designed to address the requirements of the pervasive environments, such as fault tolerance and a varying number of services and actors (Mostarda et al., 2010). To address the fault tolerance, CHOREO's semantics allows monitoring of the incoming and outgoing messages. To address the varying number of services and actors, CHOREO implemented a set-based invocation allowing the invocation of a large number of services offering the same operation.

Choreographed or peer-to-peer interactions in mobile ad-hoc networks are enabled by extending the Business Process Execution Language with additional attributes (Zhang et al., 2008). These attributes capture information about the location of participants in the choreography scenario and enabling peer-to-peer or choreographed interaction between their services.

The presented studies reveal that the current CMLs used in the ES domain relate their language construct with distinct development aspects they aim to support. The problem with this approach is that it tightly couples CMLs to a particular area of application where that development aspects is relevant. In other application areas, which emphasize other development aspects, the expressiveness of these CMLs is reduced, which makes them only partially applicable.

To address the identified problem, the goal of this study was to produce knowledge for the design of a comprehensive CML capable of supporting the choreography-relevant development aspects that are typical in ES development. Based on this goal, the following research question was derived:

*RQ: How can the expressiveness of a CML be increased to enable its applicability in the embedded systems domain?*

The answer to this RQ is formed of two parts. The first part consists of identifying the choreography-relevant development aspects in the ES domain, their articulation as design requirements, and the selection of technologies for language implementation. The second part consists of evaluating the feasibility of the derived design requirements and implementation technologies. The feasibility of the design requirements was evaluated by redesigning an existing CML based on the derived requirements. The feasibility of the selected technologies for language implementation was evaluated by implementing a prototype language editor using those technologies. It should be noted that in the rest of the text, the terms design and redesign of the CML will be used interchangeably.

The research approach, sources and analysis methods used in this study are presented in Section 2. Results of the analysis consisting of design requirements and the implementation technology proposals are presented in Section 3 and 4. Section 5 describes the feasibility evaluation of the design requirements. Section 6 concludes the study and presents the future work.

## 2 RESEARCH METHOD

This study was part of a larger research endeavour conducted during the AMALTHEA research project (AMALTHEA, 2013), whose goal was to design and empirically validate a CML that is applicable in model-driven ES development. Since the design was the focus of this research endeavour, a framework for design science research (Hevner et al., 2004) was selected to coordinate separate research efforts conducted during the project and to guide them towards their common goal. Research methods used in this study and how they are related to the results of other studies conducted in the AMALTHEA project will be described with consideration to the selected framework. Therefore, a short explanation of the framework will be presented first, while the details regarding the knowledge sources and analysis methods will follow.

### 2.1 Framework for Design Science Research

The framework for design science research consists of three major parts – the environment, knowledge base and design research (Hevner et al., 2004). The environment defines the practical problem space where the phenomenon of interest resides, and the knowledge base provides existing theories, methods and technologies which are used in the design research where the artefact is built and evaluated. In Figure 1, these three parts of the framework are presented as grey rectangles, while the knowledge sources, analysis methods and results of this study are emphasized with a black colour. The relation between the design science as an overall research framework and the work in this study is described according to the three research cycles – relevance, rigor and design cycle (Hevner, 2007). These cycles reveal the underlying logic of the framework and are therefore seen as suitable for guiding this description.

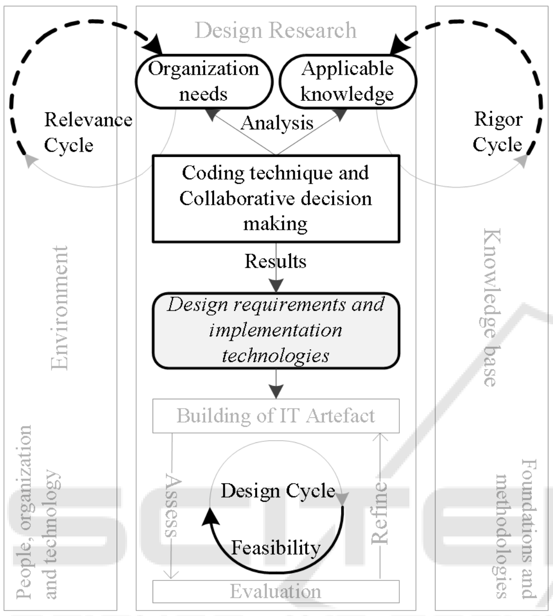


Figure 1: Design science research framework and the regulatory cycles (adopted from (Hevner, 2007)).

*The relevance cycle:* relates the practical problems identified in the environment with the design research. The upper part of this cycle, represented as a dashed arrow in Figure 1, represents the exploration of the environment, which provides various needs regarding the phenomenon being addressed with the design. The lower part of this cycle denotes the evaluation of the designed artefact in the environment.

*The rigor cycle:* relates the existing knowledge base with the design research. The upper part of this cycle, represented as a dashed arrow in Figure 1, represents the exploration of the knowledge base which provides existing theories, methods, experiences which are applicable during the design. The lower part of this circle denotes the new knowledge that is synthetized during the design and returned to the knowledge base.

*The design cycle:* represents the core of the design science research framework since it iterates between building (upper part of the circle in Figure 1) and evaluation (lower part of the circle in Figure 1) activities, which are the two main activities where the artefact is being developed. These activities, according to (Hevner, 2007), rely on the design requirements which are derived based on the knowledge sources provided by the relevance and rigour cycles.

The derivation of design requirements for the CML and proposals for the implementation technologies represent the main results of this study. These results were provided by analysing the knowledge sources provided by the relevance and rigor cycle. In Figure 1, these knowledge sources are presented as *organization needs* and *applicable knowledge*. The analysis of these knowledge sources relied on two analysis methods. The first analysis method was the *coding* which is the commonly used technique in qualitative studies. The second method was *collaborative decision making*. In Figure 1, the analysis methods and the results are presented with the rectangle and the highlighted rounded rectangle.

## 2.2 Knowledge Sources

Knowledge sources used during this study consisted of data and results from two previously conducted studies in the AMALTHEA research project (Taušan et al., 2014; Taušan et al., ). The analysis conducted in these studies resides in the upper parts of the relevance and rigor circle and represents an exploration of the environment and the knowledge base. The data and the analysis results therefore represent the organization needs and the applicable knowledge that was used in this study.

*Organization needs* were introduced to design research from the relevance cycle, and represent the exploration of the environment. These needs consisted of practical development challenges that can be addressed with choreography modelling and represent the results of the separate study presented in (Taušan et al., 2014). The data analysed in this separate study was collected by means of a) interviews and b) a review of various company specific documents and technical reports compiled during the AMALTHEA research project. Since this data was also used for the derivation of design requirements in this study, a concise description of the data is presented.

The interviews were conducted during the first quarter of 2012. Interviewees were five software architects who worked both in large, and small enterprises and who had between 10 and 26 years of experience in software development. Semi-structured interviews were used to enable interviewees to thoroughly express their views on the topics that were addressed in the questions. In addition to the interview data, the company documents were analysed as well. These documents included templates, process and work descriptions, and requirements examples. Finally, the technical reports from AMALTHEA represented a valuable source of information, especially during the compilation of the proposals for the implementation technologies.

*Applicable knowledge* was introduced to design research from the rigor cycle and represent the exploration of the knowledge base. This knowledge source consisted of a) the results and analysis of the published literature on choreography in the ES domain and b) the review of technologies for the implementation of modelling languages and the candidate languages suitable for redesign. The collection and analysis of the published literature were conducted in a separate study (Taušan et al., ) using the systematic literature review method. This study included 38 primary studies which focussed on choreography in the ES domain, while their analysis resulted in comprehensive characterizations of the choreography utilization in the ES domain. These results formed a solid theoretical base for deriving design requirements by revealing various development aspects alongside which the choreography was used, choreography specification types, the modelling languages used for the specifications and development tools. The review of potential implementation technologies and existing languages were realized by consulting the scientific literature known to researchers, exploring websites of large technology vendors and by following the discussions on various online forums. The results of this review were used during the selection of the implementation technology and language that is most suitable for the CML redesign.

### 2.3 Analysis Methods

The knowledge sources provided along the relevance and rigor circles were analysed to produce the knowledge necessary for the design of the CML which is applicable in the ES domain. Two analysis methods were used for this purpose – the coding technique and the collaborative decision-making practice. The application of these analyses methods were situated in the design research part of the research framework presented in Figure 1.

The coding technique was applied to the collected knowledge sources, with the aim of deriving the design requirements. This technique is a qualitative analysis technique in which pieces of text about the phenomenon that is studied are assigned to a code (Miles and Huberman, 1994). The code is commonly named with two to three words that summarize what is in the coded text, while the coded text represents the information base for understanding the studied phenomenon and for deriving interpretations. In this study, the organization needs and applicable knowledge represented the sources that were coded. These codes aimed to capture which ES development aspects should be supported with CML language constructs. The analysis of the coded text revealed various commonalities which were expressed with the new set of codes. Researchers then coded the knowledge sources again using the new set of codes, analysed the coded text and derived the design requirements. The coding and the analysis tasks were facilitated with NVivo tool, which is specialized software for qualitative analysis (QSR-International, 2014).

The practice of collaborative decision making was applied to select the technology for modelling language implementation and to select the existing CML that is suitable for redesign. This practice denotes joint discussions during which industry experts and researchers use their personal expertise to reason and to collaboratively decide which technology and which modelling language is the most suitable for the redesign of CML and the implementation of the CML editor. During the decision-making process, experts and researchers considered various viewpoints, such as the availability of implementation technologies, language acceptance and usability. Considering all these aspects ensured the optimality of the decisions which were made.

## 3 DESIGN REQUIREMENTS

The analysis of the data using the coding technique resulted in a set of six design requirements for CML. Each requirement focuses on a single ES development aspect and clarifies its relation with the CML. The description of each design requirement is structured of three parts – context, problem and knowledge. Context describes the observed development aspect and its environment. The problem part characterizes the observed aspect in the ES domain and articulates the identified misalignment as a design requirement. The knowledge part introduces studies which focus on the choreography use in the ES domain and which contributed to the derivation of design requirements. This structure emerged during the coding of knowledge sources and represents the commonalities that were identified.

### 3.1 Constraint-based Access

*Context:* Choreography participants are not necessarily participating in all choreography scenarios, nor do they need to interact with all other participants. For interaction to happen, participants should be aware of each other and be able to grant or to constrain the access to their resources based on agreements. *Problem:* Constraint can be described using different formats, such as condition-action, event-condition-action or decision tables. The information about the chosen format should be visible in the CML, and constraints expressed using that format should be related to the corresponding participants. This aspect is addressed with the following design requirement:

*DR1: CML should capture the format of participant's access constraints in a structured way.*

*Knowledge:* Constraining the access to participants' services has been considered in several studies. The software system is seen as a set of concurrently interacting actors (or participants), each offering a set of services in (Paech, 1997). In their view, the actor or participant, as a service owner, should allow access to its services only to participants with whom it must interact and constrain others. In (Ferrari et al., 2006), the access constraints is addressed by implementing in/out rules into their CML. Finally, in (Niemöller et al., 2011), access constraints were implemented to enable the dynamic telecommunication service selection.

## 3.2 Ad-Hoc Networks in the Choreography Scenario

*Context:* Interacting participants in a choreography scenario exchange messages through the network. Participants can therefore be seen as network nodes which are assumed to be always connected, available and ready to process an incoming message. *Problem:* The network formed from participants can often be characterized as ad-hoc in an ES domain. This means that participants (or nodes) in the scenario can connect to or disconnect from the network at any point of time, while the choreography scenario is being executed. To address the ad-hoc networking aspect, a design requirement is derived:

*DR2: CML should differentiate the participants which express ad-hoc behaviour on the network.*

*Knowledge:* Ad-hoc networking of participants is addressed in (Sen et al., 2008). Participants here are tied to mobile devices and, as such, can move in and out of the area covered with the network. In the healthcare domain, ad-hoc networking is considered in (Dar et al., 2011). In this study, ad-hoc networking of participants is seen as a consequence of the need for the healthcare system to adapt to unforeseen changes, such as changes in the patient's condition.

## 3.3 Technical and Technological Heterogeneity

*Context:* In the ES domain, participants' services involved in choreography scenarios are often executed using heterogeneous technical and technological platforms. This heterogeneity includes differences in device hardware, processing capability, implementation technology, communication protocol, service interface and message formats. *Problem:* Besides being used as an analytical tool, in model-driven ES development, the choreography specification is also parsed by tools and executed by middleware. Since technical and technological data is needed for the execution, CML should be able to capture these details. Based on this need, the following requirement is derived:

*DR3: CML should capture participants' heterogeneous technical and technological details.*

*Knowledge:* The need to enable heterogeneous participants to interact in common choreography scenarios is recognized in several studies. In (Sen et al., 2008), for example, the communication protocol heterogeneity in mobile ad-hoc networks is addressed by developing a hybrid protocol that combines publish-subscribe, store-and-forward and content-based routing protocols. In (Ferrari et al., 2006; Niemöller et al., 2011), a similar solution that focuses on middleware is proposed. Both proposals use middleware features as a mediator between heterogeneous participants and allow extensions of their middleware to accommodate the upcoming novel technologies. What differentiates these approaches is how the specified participants interact through middleware. In (Ferrari et al., 2006), strict naming of the participants' service allows middleware to resolve its implementation technology and based on that to establish the interactions. In (Niemöller et al., 2011), heterogeneous participants are specified using the technology agnostic service skeleton, while execution agents mediate the interactions and, if needed, conduct the necessary translations. Finally, how different implementation of middleware features can influence CML constructs was revealed in (Taušan et al., 2013).

## 3.4 Service Invocation Variants

*Context:* Participants in a choreography scenario are exchanging messages by invoking each other's services. For invocation to occur, a requesting participant has to know the location of the providing participant. *Problem:* Technological heterogeneity has led to various service invocation mechanisms, each requesting a specific set of data to resolve the location of the providing participant. The set of data needed for the invocation should therefore be visible in the CML, and based on this need, the following design requirement is derived:

*DR4: CML should support the description of various service invocation mechanisms.*

*Knowledge:* Examples of literature sources that were consulted during this work include four studies. Two similar service invocation mechanisms which include the supplementation of the orchestration language with attributes that hold the address of the next participant in the choreography scenario were proposed in (Pedraza and Estublier, 2009; Zhang et al., 2008). This way, instead of routing the invocation through an orchestration engine, each participant knows the location of the next service and invokes it in choreographed or peer-to-peer manner. A different approach to service invocation is proposed by (Niemöller et al., 2011). Instead of capturing the address of the next service, the language enables the description of the constraints of the next service in the interaction scenario. Using these constraints, the execution environment resolves the location of the next services and establishes the interaction. Finally, how different implementations of middleware features influence to invocations is studied in (Taušan et al., 2013).

## 3.5 Real-time Execution

*Context:* The execution of an ES is often dependant on various real-time (RT) constraints. These constraints impose strict duration times (or deadlines) for services or components to process their inputs and deliver output. Choreography scenarios, which focus on interactions between participants' services, are therefore often subject to RT constraints. *Problem:* Current CMLs offer no or only partial support for capturing RT constraints, which in those cases, are heavily tailored for the particular area of application. A generic way of capturing RT constraints for choreography in the ES domain is needed, and accordingly, the following design requirement is derived:

*DR5: CML should capture the real-time information needed for executing the choreography scenario in the ES domain.*

*Knowledge:* Three studies formed the theoretical base for studying RT requirements in CML. How a specification written in RT-UML can be translated into Choreography Description Language containing RT information is presented in (Cambronero et al., 2006). RT information expressing the time during which one participant use another's resources is explicitly supported in Scribble, which is a CML proposed in (Hu et al., 2013). Finally, (Bond et al., 2009) emphasized the importance of RT requirements in the telecommunication systems development domain.

## 3.6 Supplementary Information of CML Constructs

*Context:* CML constructs that are used to specify a choreography scenario often require additional descriptions. In most cases, these descriptions are used for documentation purposes, but they also provide additional information during the specification transformation and execution. *Problem:* In the ES domain, additional descriptions of CML constructs are often read by tools or execution environments. This is possible only if the construct is described using a structure known to the tools or engines. To address this need, the following design requirement is derived:

*DR6: CML should support a structured way of describing its language construct.*

*Knowledge:* A structured description of CML constructs was motivated with the features of the SCALE, which is a language for telecommunication service interaction modelling (Joerg and Vandikas, 2010). SCALE implemented a set of rules and key-value pairs as a structure for describing its language constructs.

## 4 IMPLEMENTATION TECHNOLOGIES FOR CML

At the beginning of the CML redesign, decisions regarding the implementation technologies and the modelling language suitable for the redesign were made. These decisions were made during the collaborative decision-making practice which resulted with the selection of the Eclipse Modelling Framework (EMF) and Sirius as implementation technologies and the Business Process Model and Notation (BPMN) as the CML to be redesigned.

*EMF* is one of the major MDE initiatives developed as part of the Eclipse ecosystem and represents a modelling and code generation facility for building tools on top of the structured data model (EMF, 2014). Selection of EMF over other MDE initiatives such as (MDA, ; Kelly and Tolvanen, 2008), was influenced by industry experts who preferred Eclipse-based technology, but also by the finding from the literature which revealed that Eclipse is the preferred technology in industries where the choreography is used for ES development (Taušan et al., ).

*Sirius* is the Eclipse technology based on EMF that enables various ways of presenting (visualizing) the structured data model (Sirius, 2014). Selection of Sirius was motivated by the need to a) represent the same structured (choreography) model using different visual forms (such as tables, trees and flow-charts) depending on which stakeholder is using the model and b) by the need to quickly develop a language and tool prototype. Representations with different forms and with different levels of detail are highly emphasized by industry experts since the choreography models are used by different roles in ES development, while the quick prototyping enables shorter design and evaluation cycles.

*BPMN* is the standardized graphical modelling notation for modelling business processes (BPMN, 2011). There are several reasons why BPMN was selected as a suitable language for redesign. These reasons include the following: a) BPMN is the most commonly used language for process modelling (Chinosi and Trombetta, 2012; Aagesen and Krogstie, 2010), b) a quality evaluation of BPMN revealed that its language constructs are well suited for capturing choreographies (Cortes-Cornax et al., 2011), c) experts from companies that participated in the AMALTHEA research project have experience with BPMN and strongly support its selection, and d) BPMN is independent of implementation technologies, capable of presenting both the choreography and the orchestration point of view, and allows extensions based on the needs of a vertical market. The redesign of the existing language compared with the development of new language is also expected to ensure its quick adoption and its future use since the developers that are skilled in BPMN can easily master the redesigned language.

## 5 FEASIBILITY EVALUATION

The feasibility of the study results was evaluated by redesigning the proposed CML according to the derived design requirements and by implementing a prototype language editor using the selected technologies. This evaluation aims to a) show that the proposed CML can be redesigned in a way to realize the design requirements and b) to demonstrate the potential benefits of the implementation technologies used for the development of the language editor.

### 5.1 CML Redesign

BPMN, which is the selected CML, is redesigned by changing its meta-model, which included the addition of new classes and modifications to its existing classes by adding new attributes. The redesign of the BPMN consisted of the following four steps: a) inspection of the BPMN meta-model, b) identification of classes, c) redesign and d) summarization.

*Familiarization* with the BPMN meta-model included a thorough inspection of the BPMN's classes, their attributes and class relationships. The main purpose of this step was to understand the model's specifics and how it can be modified to accommodate the design requirements. This inspection relied on the BPMN specification document (BPMN, 2011) and the electronic version of the meta-model obtained from the Eclipse git repository (Eclipse Git repositories, 2015).

*Identification* of the classes included a joint examination of design requirements and the BPMN meta-model. The main purpose of this step was to express the design requirements as new classes and attributes and to identify the suitable classes in the meta-model that can be supplemented with the newly designed classes and attributes. New classes and attributes together with the ones identified in the meta-model are presented and described in Table 1.

*Redesign* denotes the actual changes to BPMN's meta-model, during which the newly identified classes and attributes were integrated into the meta-model. Table 1 consists of four columns in which the newly identified classes, attributes, their description and the classes from the meta-model which they supplement were presented. The rows in Table 1 that contain classes, attributes and their descriptions are lined up in a way that they border the cells in the fourth column in which the BPMN classes which they supplement are described. This description also contains a concise explanation of the relation and indicates the design requirement it realizes. Note that some of the identified attributes were first encapsulated in a dedicated class and then related to an existing BPMN class, while other attributes directly extend the existing BPMN classes. These attributes can be differentiated based on whether there is a class name in the corresponding cell in the column class.

*Summarization* included the documentation of the work done during the redesign. The main purpose of this step was to archive the data produced during the redesign that will later be used for the derivation of design knowledge.

Table 1: Mapping of newly proposed classes and attributes to BPMN language constructs.

| Class | Attribute | Description of the class or attributes | Supplemented BPMN classes |
|---|---|---|---|
| | isAdHoc: boolean | Whether the participant expresses ad-hoc behaviour | *Participant.* Represents an autonomous management authority such as organization unit or different company that is engaged in the choreography scenario. In cases when isAdhoc is set "true", Participant's behaviour is characterized with data grouped under the Motion class. Together, isAdhoc and Motion class, realize the R2 design requirement. partConstraint attribute supplements the Participant with attribute which specifies the constraint format and realize the R1 design requirement. The Participant's capability is characterized with Performance class data. This class partially fulfils the R3 design requirement. |
| | ptCriterion: Enum | Shows the criterion used to identify the participant (e.g. is it a device, smartphone, or factory machine) | |
| | ptConstraint: Enum | Reveals the format for defining constraints for invoking the participant's roles (e.g. Condition-Action) | |
| | ptMonitor: string | Location of a service for service interaction monitoring | |
| Motion | | Captures the participants' movement in the ad-hoc network | |
| | motEnt: date/time | Shows the time when the participant joined the network | |
| | motExi: date/time | Shows the time when the participant left the network | |
| | motDuration: duration | Shows the expected time, in minutes, during which the participant will be connected on the network | |
| Performance | | Characterizes the hardware capability of devices where the participant is executing | |
| | CPU: Enum | Indicates the CPU unit type | |
| | coreNo: small | Indicates the number of cores in CPU | |
| | RAM: double | Indicates the amount of RAM memory (e.g. in MB) | |
| | netType: Enum | Indicates the underlying network technology (e.g. CAN) | |
| | trRate: double | Shows data transfer rate (e.g. Mbit/s) | |
| Real-Time | | Characterises the real-time constraints | *FlowNode.* Groups all elements that can appear in the process flow. This class is related with a single RTime class to enable real-time information capture. RTime class realizes R5 design requirement. |
| | rtStar: time stamp | Start of the execution | |
| | rtEnd: time stamp | End of execution | |
| | rtDur: duration | Duration of execution | |
| | rtDelay: duration | Indicates the delay the following action | |
| | roleIface: Enum | Interface type (e.g. WSDL, CORBA IDL) | *Interface.* Defines a set of operations that are implemented by Participant's service. Proposed attributes extend this constructs and enable the specification of communication related data. Attributes partially realizes the R3 design requirement. |
| | roleProt: Enum | Communication protocol (e.g. http, IIOP) | |
| | roleMsgForm: | Message format used by the role (e.g. SOAP) | |
| | roleLoc: string | Location of the Participant or its Role on the network. (e.g. URL, IP, role name) | |
| | isPushPull: Enum | Indicates is the invocation of participant's services realized using push or pull message invocation strategy. | *MessageFlow.* Show the flow of the message between two Participants. isPushPull attribute specifies whether the participant receives the message via "push" or "pull" strategy. Attribute partially realizes the R3 design requirement. invocType extends the construct to specify the invocation type. Depending on selection, the tool enforces the specification of other attributes relevant to the selected type. The attribute realizes the R4 design requirement. |
| | invocType: Enum | Indicates the type of service invocation. Each invocation types groups a set of data the participant has to provide to invoke the needed service. | |
| Structured Description | | Enables the structured description of process elements in the scenario | *TextAnnotations.* A mean for a modeller to provide additional information for the reader of the specification. This class is related with one or more StructDesc class to realize the R6 design requirement. |
| | Key: string | Property of the element in the scenario | |
| | Value: string | Value of the property | |
| | Desc: string | Additional, free form description of the element, property and value | |

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

## 5.2 Prototype Language Editor

The prototype editor for the redesigned CML was implemented in two phases. In the first phase, the specified supplements to BPMN's meta-model were implemented using the EMF technology. In the second phase, the choreography language editor was implemented on top of the redesigned meta-model using the Sirius technology. To present the language and editor, we adopted the scenario for a central locking system from the automotive use case presented in (Krüger et al., 2004). Since this use case defines the roles and their interaction needed to automatically unlock the car door, it is seen as suitable for being presented with the choreography. The central locking system scenario, features of the language and editor, and how they support the derived design requirements are presented in Figure 2.

### 5.2.1 Central Locking System Scenario

The scenario to unlock the car doors starts when the passenger presses the button on its remote car key. In Figure 2, this is presented in the first interaction step, labelled as *Unlock initiation*, where the key sends a message to the car's central locking system for further processing. Note that in choreography, instead of displaying the actual participants' names, their roles in the scenario are displayed instead. The reason for this is that a single participant can have multiple roles which denote its contribution in that scenario. Also, if needed, the editor can easily be customized to show the participant names as well.

After the Unlock initiation step, the scenario continues on two parallel branches. Two interaction steps on the upper branch in Figure 2 show the actual unlocking of the car doors and the signalling to the passenger that the doors are open using blinkers, sound or car lights. The lower branch of the scenario shows the *Comfort adjustments*. First, the system checks if the passenger has predefined comfort preferences. If yes, the *comfort manager* interacts with *comfort controllers* to adjust, for example, seat height, rear-view mirrors, favourite radio stations or cabin temperature. If there are no comfort preferences recorded, this interaction step is skipped, and the two branches are merged to complete the scenario.

### 5.2.2 Language Editor Features

The language editor is implemented as a visual modelling editor, meaning that it enables visual (or graphical) development and the representation of choreography scenarios. This editor consists of three main areas, and these areas are the main canvas, palette and the properties area. The main canvas is the area in which users specify a choreography scenario. This is achieved by dragging and dropping choreography language constructs from the palette and by setting various properties of these constructs through the property area. In Figure 2, these areas are highlighted with red rectangles.

The identified design requirements are realized by redesigning BPMN; however, the features of the language editor also contribute to their realization. One such feature enables the specification and representation of choreography scenarios with different levels of details depending on the users' role in ES development. This is achieved using layers, which is the Sirius technology's way to relate subsets of language constructs with a concrete role. The current version of the editor implements layers for two roles —analysts and implementers. The reason for these two roles is that in the ES domain, choreography specifications can be used both for analysis and implementation (see DR3) and therefore used by these roles. A switch to navigate between the layer is highlighted in the upper left corner of Figure 2 with a red rectangle.

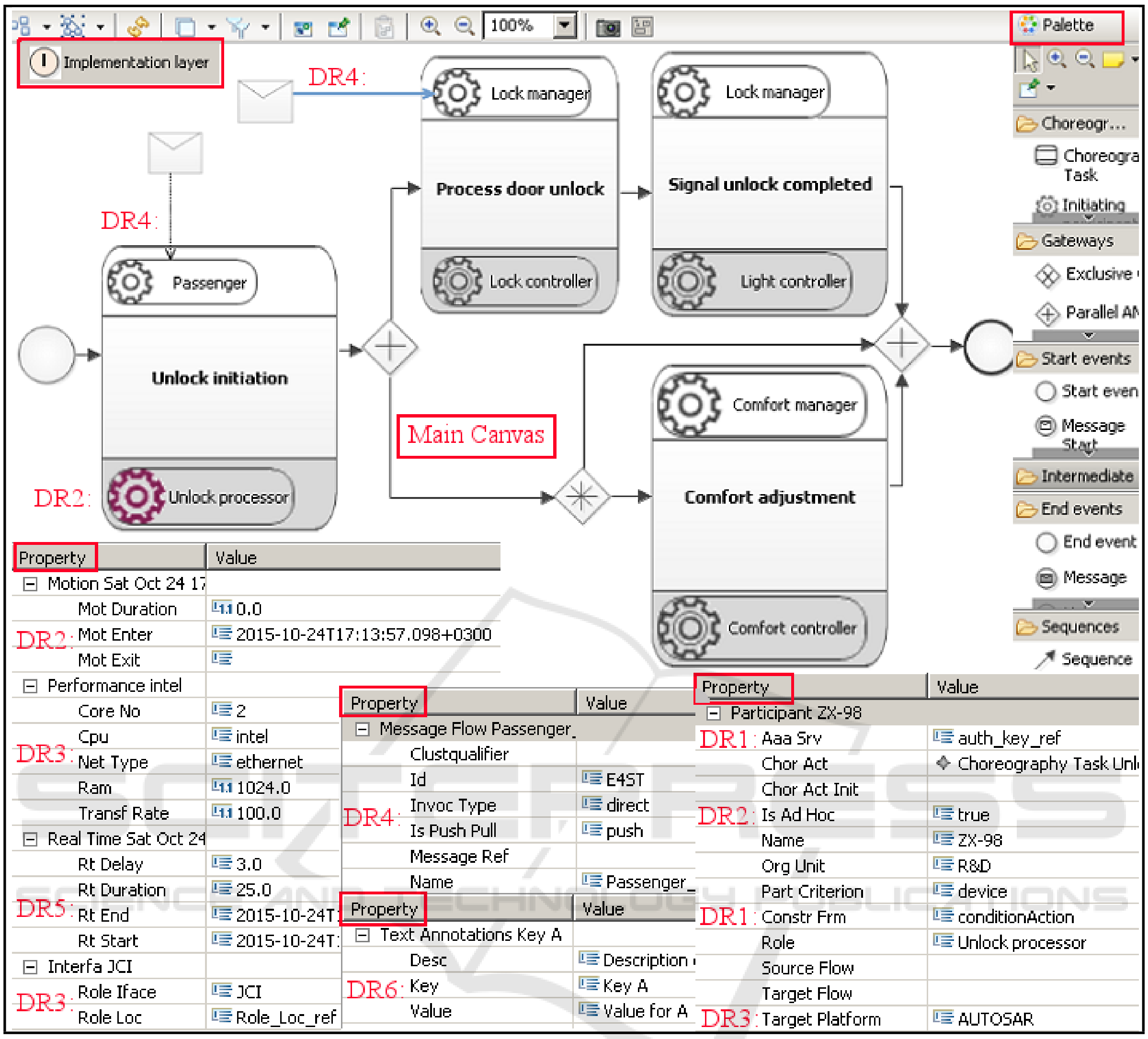
Besides the mentioned feature, the language editor also supports the developers work by: a) enabling a hierarchical organization of the choreography scenarios, b) preventing the relationships between language constructs that should not be related, and c) allowing quick customizations of language constructs visualization. For example, in the scenario presented in Figure 2, DR4 and DR6 utilize the construct visualization feature to differentiate their language construct based on their property values.

### 5.2.3 Design Requirements Support

Six design requirements for CMLs were derived and presented in this study. Based on these requirements, BPMN's meta-model was redesigned and the prototype editor was developed. How the design requirements are supported with the redesigned language and editor is presented in Figure 2 and these requirements are marked with DR1 to DR6 labels.

*DR1: Constraint-based Access.* This requirement is implemented as part of the participant language construct and allows users to record details about constraint rule types. In the presented scenario, the signal sent form the passenger's key is first authenticated using authentication, authorization, and accounting service (Aaa service), and if authenticated, the condition-action rule that allows the advancement of the scenario is executed.

*DR2: Ad-Hoc Networks in Choreography Scenario.* This requirement allows users to record whether the participant expresses ad-hoc behaviour.



| | Property | Value |
|---|---|---|
| | ⊟ Motion Sat Oct 24 17 | |
| | Mot Duration | 0.0 |
| DR2: | Mot Enter | 2015-10-24T17:13:57.098+0300 |
| | Mot Exit | |
| | ⊟ Performance intel | |
| | Core No | 2 |
| DR3: | Cpu | intel |
| | Net Type | ethernet |
| | Ram | 1024.0 |
| | Transf Rate | 100.0 |
| | ⊟ Real Time Sat Oct 24 | |
| | Rt Delay | 3.0 |
| | Rt Duration | 25.0 |
| DR5: | Rt End | 2015-10-24T |
| | Rt Start | 2015-10-24T |
| | ⊟ Interfa JCI | |
| DR3: | Role Iface | JCI |
| | Role Loc | Role_Loc_ref |

| | Property | Value |
|---|---|---|
| | ⊟ Message Flow Passenger_ | |
| | Clustqualifier | |
| | Id | E4ST |
| DR4: | Invoc Type | direct |
| | Is Push Pull | push |
| | Message Ref | |
| | Name | Passenger_ |

| | Property | Value |
|---|---|---|
| | ⊟ Text Annotations Key A | |
| | Desc | Description |
| DR6: | Key | Key A |
| | Value | Value for A |

| | Property | Value |
|---|---|---|
| | ⊟ Participant ZX-98 | |
| DR1: | Aaa Srv | auth_key_ref |
| | Chor Act | Choreography Task Unl |
| | Chor Act Init | |
| DR2: | Is Ad Hoc | true |
| | Name | ZX-98 |
| | Org Unit | R&D |
| | Part Criterion | device |
| DR1: | Constr Frm | conditionAction |
| | Role | Unlock processor |
| | Source Flow | |
| | Target Flow | |
| DR3: | Target Platform | AUTOSAR |

Figure 2: Central locking system scenario, language and editor features.

If yes, that participant is visually diversified in a diagram using different colour, and users can characterize its motion by recording the motion details in property fields. In the presented scenario, the Unlock processor is marked to express the ad-hoc behaviour. The reason for this behaviour is that the interaction between the Passenger (signal sender) and car (signal receiver) can be hampered due to the distance, interfering objects or lack of power supply.

*DR3: Technical and Technological Heterogeneity.* This requirement is implemented as part of the participant language construct and allows users to record the participant's target platform and to characterise its hardware. The performance is characterised by recording details such as the CPU, numbers of CPU cores, amount of random access memory and the target platform. As a demonstration, recorded technical and technological details for the Unlock processor are presented in Figure 2

*DR4: Service Invocation Variants.* This requirement is implemented as part of the message flow language construct which relates the participants and the message being exchanged and enables visual differentiation based on the service invocation variant. Two variants and their visualization are presented in Figure 2. In the first variant, the Passenger sends a message by directly invoking the Unlock processor service. In the second, the Lock manager invokes the Lock controller using the platform's messaging service.

*DR5: Real-time Execution.* This requirement is implemented as part of the participant construct and the message flow constructs. In the case of participants, these properties show the time needed to process the request, while in the case of message flows, the properties characterises the time elapsed between message sending and receiving. In the presented scenario, the duration of a participant's processing time is limited to 25 milliseconds.

*DR6: Supplementary Information of Language Constructs.* This requirement is implemented as part of each language construct and enables additional descriptions of the construct. The selection of syntax for expressing key-value pairs is up to the users' preferences. The key and its value can be in the form of a simple variable, and its score (as it is in the presented example) can be regular expressions or involve more complex syntax.

## 6 CONCLUSION AND FUTURE WORK

The problem on which the study focused was the limited expressiveness of CMLs caused by the tight coupling of their language construct with distinct development aspects. To increase the expressiveness of CMLs, a set of six design requirements were introduced, and suitable language implementation technologies were suggested. The introduced design requirements represent the embodiment of different needs which are typical in ES development and which can be addressed with the choreography modelling. Language implementation technologies and the BPMN, as the language selected for the redesign are fully applicable in the MDE context and represent our proposals for the implementation of CML in the ES domain.

The feasibility of these results was evaluated by redesigning the selected CML according to design requirements and the implementation of the language editor with the suggested technologies. The results of this evaluation were promising, indicating that the design requirements can be implemented on top of the selected CML and that they increase the expressiveness of that CML. This feasibility evaluation represents a step towards the implementation of a comprehensive CML that is fully applicable in the ES domain, where choreography is increasingly used. Besides the increase in expressiveness, redesigned CML includes additional favourable properties, such as its suitability for model-driven development (due to the implementation based on the meta-model), easier comprehension of the specification (due to visual representations in the language editor), and the adaptability of the language according to the target middleware platforms and the specific ES development context.

*Future work* will focus on evaluating the CML wiht industry experts and further developing the language and editor. Further development of the editor will focus on the automation of tasks, such as code generation based on underlying middleware platforms.

## ACKNOWLEDGEMENTS

This study was part of the ITEA 2 AMALTHEA international research project and it is supported by TEKES. The authors are grateful for their support and cooperation.

## REFERENCES

Aagesen, G. and Krogstie, J. (2010). Analysis and design of business processes using BPMN. In *Handbook on Business Process Management 1*, pages 213–235. Springer.

AMALTHEA (2013). AMALTHEA.

Barros, A., Dumas, M., and Oaks, P. (2005). A critical overview of the web services choreography description language. *BPTrends Newsletter*, 3:1–24.

Bond, G., Cheung, E., Fikouras, I., and Levenshteyn, R. (2009). Unified telecom and web services composition: problem definition and future directions. In *Proceedings of the 3rd International Conference on Principles, Systems and Applications of IP Telecommunications*, page 13. ACM.

BPMN (2011). Business Process Model and Notation Version 2.0.

Cambronero, M.-E., Diaz, G., Pardo, J. J., Valero, V., and Pelayo, F. L. (2006). RT-UML for modeling real-time web services. In *Services Computing Workshops, 2006. SCW'06. IEEE*, pages 131–139. IEEE.

Cannata, A., Gerosa, M., and Taisch, M. (2008). A Technology Roadmap on SOA for smart embedded devices: Towards intelligent systems in manufacturing. *Industrial Engineering*.

Chinosi, M. and Trombetta, A. (2012). BPMN: An introduction to the standard. *Computer Standards & Interfaces*, 34(1):124–134.

Ciancia, V., Ferrari, G., Guanciale, R., Strollo, D., and Tuosto, E. (2011). Model-driven development of long running transactions. In *Rigorous software engineering for service-oriented systems*, pages 326–348. Springer.

Cortes-Cornax, M., Dupuy-Chessa, S., Rieu, D., and Dumas, M. (2011). Evaluating choreographies in bpmn 2.0 using an extended quality framework. In *Business Process Model and Notation*, pages 103–117. Springer.

Dar, K., Taherkordi, A., Rouvoy, R., and Eliassen, F. (2011). Adaptable service composition for very-large-scale internet of things systems. In *Proceedings of the 8th Middleware Doctoral Symposium on - MDS '11*, pages 1–6, New York, USA. ACM Press.

Dijkman, R. and Dumas, M. (2004). Service-oriented design: A multi-viewpoint approach. *International journal of cooperative information systems*, 13(04):337–368.

Eclipse Git repositories (2015). org.eclipse.bpmn2.git - MDT BPMN2 metamodel.

EMF (2014). Eclipse Modeling Framework Project.

Ferrari, G., Guanciale, R., and Strollo, D. (2006). Jscl: A middleware for service coordination. In *Formal Techniques for Networked and Distributed Systems-FORTE 2006*, pages 46–60. Springer.

Gilart-Iglesias, V., Macia-Perez, F., Marcos-Jorquera, D., and Mora-Gimeno, F. J. (2007). Industrial Machines as a Service: Modelling industrial machinery processes. In *2007 5th IEEE International Conference on Industrial Informatics*, volume 2, pages 737–742. IEEE.

Hevner, A. R. (2007). A Three Cycle View of Design Science Research A Three Cycle View of Design Science Research. *Scandinavian journal of information systems*, 19(2):87–92.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 28(1):75–105.

Hu, R., Neykova, R., Yoshida, N., Demangeon, R., and Honda, K. (2013). Practical interruptible conversations. In *Runtime Verification*, pages 130–148. Springer.

Joerg, N. and Vandikas, K. (2010). SCALE – A language for dynamic composition of heterogeneous servicesle.

Kaur, N., McLeod, C. S., Jain, A., Harrison, R., Ahmad, B., Colombo, A. W., and Delsing, J. (2013). Design and simulation of a SOA-based system of systems for automation in the residential sector. In *Industrial Technology (ICIT), 2013 IEEE International Conference on*, pages 1976–1981. IEEE.

Kelly, S. and Tolvanen, J.-P. (2008). *Domain-specific modeling: enabling full code generation*. John Wiley & Sons.

Krüger, I. H., Gupta, D., Mathew, R., Moorthy, P., Phillips, W., Rittmann, S., and Ahluwalia, J. (2004). Towards a process and tool-chain for service-oriented automotive software engineering. *Architecture*, 2:R2.

Liggesmeyer, P. and Trapp, M. (2009). Trends in embedded software engineering. *Software, IEEE*, 26(3):19–25.

MDA. OMG Model Driven Architecture.

Miles, M. B. and Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook*. Sage.

Mostarda, L., Marinovic, S., and Dulay, N. (2010). Distributed orchestration of pervasive services. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 166–173. IEEE.

Niemöller, J., Vandikas, K., Levenshteyn, R., Schleicher, D., and Leymann, F. (2011). Towards a service composition language for heterogeneous service environments. In *Intelligence in Next Generation Networks (ICIN), 2011 15th International Conference on*, pages 121–126. IEEE.

Paech, B. (1997). A framework for interaction description with roles. Technical report, TECHNISCHE UNIVERSITAT MUNCHEN - INSTITUT FUR INFORMATIK, Muncen.

Pedraza, G. and Estublier, J. (2009). Distributed orchestration versus choreography: The focas approach. In *Trustworthy Software Development Processes*, pages 75–86. Springer.

Peltz, C. (2003). Web services orchestration and choreography. *Computer*, 36(10):46–52.

QSR-International (2014). NVivo 10 research software for analysis and insight.

Schmidt, D. C. (2006). Model-driven engineering. *COMPUTER-IEEE COMPUTER SOCIETY-*, 39(2):25 – 31.

Scholz, A., Gaponova, I., Sommer, S., Kemper, A., Knoll, A., Buckl, C., Heuer, J., and Schmitt, A. (2009). ∈ SOA-Service Oriented Architectures adapted for embedded networks. In *7th IEEE International Conference on Industrial Informatics, 2009. (INDIN)*, pages 599–605, Cardiff. IEEE.

Sen, R., Roman, G.-C., and Gill, C. (2008). Cian: A workflow engine for manets. In *Coordination Models and Languages*, pages 280–295. Springer.

Sirius (2014). Sirius - The easiest way to get your own Modeling Tool.

Starke, G., Kunkel, T., and Hahn, D. (2013). Flexible collaboration and control of heterogeneous mechatronic devices and systems by means of an event-driven, SOA-based automation concept. In *Industrial Technology (ICIT), 2013 IEEE International Conference on*, pages 1982–1987. IEEE.

Taušan, N., Aaramaa, S., Lehto, J., Kuvaja, P., Markkula, J., and Oivo, M. (2014). Customized Choreography and Requirement Template Models as a Means for Addressing Software Architects' Challenges. In *ICSEA 2014, The Ninth International Conference on Software Engineering Advances*, pages 55–63, Nice, France. IARIA XPS Press.

Taušan, N., Lehto, J., Kuvaja, P., Markkula, J., and Oivo, M. (2013). Comparative Influence Evaluation of Middleware Features on Choreography DSL. In *ICSEA 2013, The Eighth International Conference on Software Engineering Advances*, pages 184–193. IARIA XPS Press.

Taušan, N., Markkula, J., Kuvaja, P., and Oivo, M. Choreography in Embedded Systems Domain: A Systematic Literature Review. Manuscript submitted for publication.

Zhang, Z., Sun, W., Chen, W., Mao, D., and Xu, Y. (2008). An extended composite service choreography language for decentralized execution in MANETs. In *Communication Technology, 2008. ICCT 2008. 11th IEEE International Conference on*, pages 592–595. IEEE.