

Noname manuscript No.
(will be inserted by the editor)

Artefact-based Requirements Engineering: The AMDiRE Approach

D. Méndez Fernández · B. Penzenstadler

Received: date / Accepted: date

Abstract The various influences in the processes and application domains make Requirements Engineering (RE) inherently complex and difficult to implement. In general, we have two options for establishing an RE approach: we can either establish an activity-based RE approach or we can establish an artefact-based one where project participants concentrate on the RE artefacts rather than on the way of creating them. While a number of activity-based RE approaches have been proposed in recent years, we have gained much empirical evidence and experiences about the advantages of the artefact-based paradigm for RE. However, artefact orientation is still a young paradigm with various interpretations and practical manifestations whereby we need a clear understanding of its basic concepts and a consolidated and evaluated view on the paradigm.

In this article, we contribute an artefact-based approach to RE (AMDiRE) that emerges from six years of experiences in fundamental and evidence-based research. To this end, we first discuss the basic notion of artefact orientation and its evolution in recent years. We briefly introduce a set of artefact-based RE models we developed in industrial research cooperations for different application domains, show their empirical evaluations, and their dissemination into academia and practice, eventually leading to the AMDiRE approach. We conclude with a discussion of experiences we made during the development and different industrial evaluations, and lessons learnt.

Keywords Requirements Engineering · Artefact Orientation · Empirical Evaluation

Daniel Méndez Fernández
Technische Universität München, Garching
Tel.: +49-89-28917056
E-mail: mendezfe@in.tum.de

Birgit Penzenstadler
University of California, Irvine
E-mail: bpenzens@uci.edu

1 Introduction

Requirements Engineering (RE) is an important success factor for software and systems development projects as precise requirements are critical determinants of quality [9]. Although the discipline is known to be crucial for the success of every project, we still observe companies struggling with their RE process. Many of these companies have unclear roles and responsibilities but a detailedly defined process that is obligatory for all projects. RE is too often performed mindlessly or even faked [50], without awareness of the reasons why a process step should (or should not) be executed, and without awareness of how to structure and specify the results [45].

A major reason for this circumstance is that many things are not clear from the beginning of a project, which makes the discipline inherently complex and volatile. The need for flexibility is additionally hardened by potentially large amounts of requirements [57], which are too often insufficiently structured in spreadsheets. The effects of this circumstance can be often observed in incomplete and inconsistent requirements, and, finally, in failed projects.

The *chaos report* from the Standish Group [66] states that 44 % of the reasons for failed projects have their origin in insufficient RE. As the report takes only a limited view into RE itself and is also known to have serious flaws in its design negatively affecting the validity of the results [18], we launched a series of empirical investigations on practical problems in RE and how those problems manifest themselves in the whole software development process [40, 44]. We discovered that the missing awareness of what should be done in RE manifests in irreproducible, incomplete, and inconsistent artefacts without clear terminology, all together seen to be the major reason for time overruns, cost overruns, and eventually for failed projects [40, 43]. A solution to these problems is to establish a company-wide *RE reference model* that should support

1. flexibility in the way of working to cope with the various influences in individual project environments, and
2. the reproducible creation of resilient and detailed specification documents.

In Zave's classification of research efforts in RE, this addresses the two problems of integrating multiple views and representations, and obtaining complete, consistent, and unambiguous specifications [74, p.317].

There are two basic paradigms for the establishment of such an RE reference model: activity orientation and artefact orientation. Activity orientation means to define the reference model by means of detailed interconnected procedures that dictate which methods to combine and use in which project situation [67]. The underlying idea is to define a situation-specific process by a set of small steps, i.e. methods to be performed in a particular order to create certain artefacts as outcome (see also [8, 68]). In contrast, artefact orientation establishes a blueprint of the created RE results, their contents, and their dependencies [41]. That is, we abstract from the way of creating the results by the use of particular methods and modelling notations and specify *what* has to be done rather than dictating *how* to do something.

In our experience, the focus on RE artefacts strongly supports achieving the goals of a flexible process that still leads to detailed and, to some extent, (semantically) accurate RE specifications [41, 42]. Our process-agnostic focus on what should be created in a project in contrast to how to do something allows us to abstract from the variability in the processes, because the actual creation of artefacts by the use of particular methods in a particular sequence is reduced to the created artefacts, their contents, and their dependencies, all defined in the artefact-based reference model of a company [41, 42].

Problem Statement. Although we have made first steps into the direction of gathering a common understanding about artefact orientation [41, 45, 42], the paradigm is still young and it comes too often with various interpretations and manifestations in practice. In fact, little is yet known about how to establish an artefact-based RE approach in practice, which basic concepts have to be taken into account during this establishment, and what benefits as well as shortcomings the paradigm brings. This is, however, crucial to steer further evidence-based research within the various research communities and to increase the awareness of the basic principles for the practical application of the paradigm.

Objectives. In this article, we aim at providing a consolidated and empirically evaluated view on artefact-based requirements engineering.

Contribution. To provide a consolidated view on artefact-based requirements engineering, we contribute a domain-independent, artefact-based RE approach (the AMDiRE approach), which emerged from six years of experiences in fundamental and evidence-based research. Our contributions are intended to serve more than one purpose:

1. We introduce the basic concepts of artefact orientation in RE that we have established in fundamental research devoted to this area to lay a terminological and conceptual foundation.
2. We introduce our 6 years of research projects and resulting artefact-based RE approaches to support a common understanding of the various concepts and different interpretations of artefact orientation disseminated into academia and practice.
3. We contribute an artefact-based approach to RE, which uses a tailorabile artefact model for domain-independent RE (AMDiRE) as its backbone, and which consolidates our previously developed and evaluated approaches.
4. We share our experiences in the development of artefact-based RE approaches, lessons learnt, and conducted empirical evaluations in industrial contexts. The evaluations also show the practical implications that the different interpretations of artefact orientation have in practice.

With our contributions, we aim at supporting researchers as well as practitioners: Researchers can directly build their fundamental, educational, and evidence-based work upon our artefact model and our experiences to steer

their research in a problem-driven manner. Practitioners can directly apply our model in their own socio-economic contexts with the awareness of the benefits and shortcomings of the incorporated concepts.

Delimitations. Instead of preaching the use of one paradigm while neglecting potential benefits of the other, it is our intention to clarify the notion of artefact orientation in RE, draw an outline of its practical application, and discuss the lessons we learnt in recent years. We therefore also discuss the evolution of the paradigm from our experience, and contribute a consolidated approach as a result of various industrial research cooperations. We do not intent to propagate the dogmatic application of artefact orientation for all domains, nor do we claim its valid advantages to hold for all purposes. In fact, we agree with Tell and Babar [67] that, on the long run, we can make use of the benefits of both paradigms while limiting their shortcomings.

Research Method. Our contribution at hand, in particular the AMDiRE approach, emerges from a series of different artefact-based RE reference models developed in different research cooperations. For each development in a specific socio-economic context, we followed the principles of empirical design science [70, 72], i.e. we applied scientific methods in practical contexts to establish an artefact-based RE approach in response to company-specific problems and goals (see also Sect. 5.1 where we discuss our general experiences in the construction of those models as well as the procedure we followed). In each project, we started with a problem analysis (see, e.g., [43]) to infer a set of improvement goals, before developing particular artefact models via technical action research workshops [71] with our partners from industry. We finally conducted case study research to evaluate each of the developed artefact models w.r.t. the previously determined improvement goals and investigated to what extent we solved the discovered problems. This allowed us to get a deeper understanding on the various characteristics artefact-based approaches can have in dependency to various goals, and what implications those characteristics have when applying the models in practical environments. In Fig. 1, we depict the procedure on the left side.

In a second step, we consolidated the results of the various development and evaluation procedures and synthesised the key concepts of the created artefact models into the AMDiRE approach, which forms the main contribution of this article (see the middle side of Fig. 1). So far, we see the resulting AMDiRE approach to be successful, because up to now the concepts from which we inferred AMDiRE have resulted in successful (evaluated) RE reference models leading, e.g. to new company-specific RE standards (see, e.g. the BISA approach in Table 2, Sect. 3.2).

Some of the former models and their empirical evaluations have been published earlier [41, 42]. This article presents the consolidation of our work and the actual resulting AMDiRE model on the basis of its development over time.

In contrast to the empirical evaluations of our previous artefact models where we conducted comparative case studies to evaluate to what extent the

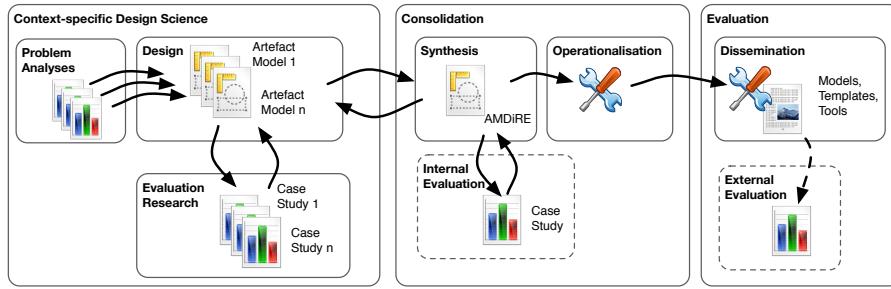


Fig. 1: The applied research method.

developments lead to an improvement of previously used activity-based RE approaches, we do not provide such evaluation for AMDiRE in this paper. The reason is that we reached the point where it yet has to be shown whether our approach can be used by others if we are not involved at all, thus, we need an external evaluation independently carried out by unbiased researchers and practitioners not involved in the development of AMDiRE. For this reason, we make our contribution and its operationalisation (e.g. relating models, tools, and evaluation templates) openly accessible [37] and disseminate our results from 6 years of research with the article at hand. This lays the foundation for the final external evaluation, depicted on the right side of Fig. 1.

Outline. The remainder of the article is as follows. In Sect. 2, we discuss the work directly related to our contributions, and the gaps we intend to close. We conclude with a discussion of the fundamentals in artefact orientation and the terminology we use in context of this article. In Sect. 3, we then introduce the background of artefact orientation resulting from our fundamental, conceptual, and empirical work in this area, i.e. introduce the previously developed artefact models and give a first introduction into the different case studies we conducted with those models. After discussing the synthesis of the models in Sect. 3.3, we present the AMDiRE approach in detail in Sect. 4. In Sect. 5, we finally discuss our experiences, our evaluations, and the lessons learnt, and conclude with Sect. 6.

2 Fundamentals and Related Work

We first discuss the areas of activity orientation and then the fundamentals in artefact orientation as well as the terminology used in context of this article.

2.1 From Activity Orientation to Artefact Orientation

Activity orientation is based on the idea of providing an RE reference model as an ordered set of activities and methods, each defining procedures and

techniques for a particular purpose [48], from which project participants can select the appropriate one to design their project-specific RE process. Each activity, e.g. how to apply use cases [17], is performed by a particular role that creates the corresponding artefact type, e.g. the requirements specification. Each of those techniques is then placed into a particular sequence of application and used to specify the RE results [6].

At the organisational level, these activity oriented RE reference models are integrated into activity-based software process models that, for example, rely on the *Software & Systems Process Engineering Meta-Model* (SPEM) [49], such as the *Rational Unified Process* (RUP) [33]. Approaches that provide means to systematically select and combine methods at project level are addressed, in turn, by the research area of *Situational Method Engineering* [8, 68]. This area can be complemented by (*content-centric*) *Decision Support Systems* [56], which contribute approaches to select, classify, and rate a set of alternatives in the choice of methods (and description techniques) according to project parameters.

Although the importance of a well-defined artefact model is recognised in the area of activity orientation [19], the definition of artefacts, their contents, and especially their dependencies is not in scope of available approaches. Braun et al. [6] discovered that only 50% of the analysed approaches include an artefact description at all, while the other 50% reduce the artefacts to an outcome of self-contained and interconnected methods that produce the artefacts. A first contribution that addresses the incorporation of artefacts into those activity-centric software processes is made by Silva and Oliveira [64] who propose a concept of meta-modeling to define an artefact layer and a process layer for a better organisation of software artefact authoring. They illustrate their approach with a use case specification outline, but do not yet provide a complete artefact model or reference implementation that would provide insights into strengths and weaknesses [34].

Considering the absence of strong empirical work in the area of activity orientation [51] and, thus, following a purely argumentative line of reasoning, activity-oriented approaches still have difficulties to overcome the problem of providing a means to support a flexible RE process that guides the creation of consistent RE artefacts. In contrast, when following the principles of *artefact orientation*, we are supposed to define an RE reference model by defining the artefacts, their contents, and their dependencies rather than dictating the way of creating the artefacts, thus, supporting flexibility in the process and the creation of detailed, consistent RE artefacts. First evidence for the benefits of artefact orientation is provided by industrial case studies that evaluate both paradigms in a comparative manner, e.g. [42] (see also Sect. 3).

The basic idea of artefact orientation is, however, not new. First artefact models have been proposed as part of checklists and templates for RE, for example, with the VOLERE requirements specification templates [60] or the IEEE recommended practice for software requirements specifications (IEEE std. 830-1998) [26]. Those templates provided a first, common understanding on the general contents to be considered in RE artefacts in the form of generic

tables of content, but they did not consider the dependencies within and between the contents. The latter is, however, important to support syntactically consistent result structures.

First content-related dependencies resulting from refinement and decomposition in the modelling concepts are provided by Berenbach et al. [2, chp. 2]. These cover the basic concepts previously developed in a research co-operation between Siemens Corporate Research and Technische Universität München (TUM) [21] (see also Sect. 3). They provide an RE artefact model and name the key components for measurable RE artefacts, include a first process guideline, and suggest practices for their elaboration.

This and similar artefact models enable an understanding about how to structure RE artefacts and how the contents relate to each other. However, those models are limited to general content descriptions rather than providing clear definitions of the modelling concepts used, for example, to create use case models. Thus, they still do not support syntactically consistent result structures.

This non-exhaustive list of artefact-based approaches already shows that we, as a research community, have developed different views on artefact models depending on their intended purpose. More structure-oriented artefact models, like the one provided by Berenbach et al. [2, chp. 2], allow for a clear process integration, since a simplified view on the contents of the artefacts can be integrated with process elements like milestones. More content-oriented artefact models, like the one provided by Schaetz et al. [63], focus on (tool-supported) seamless modelling, although a process integration becomes difficult due to the increased complexity in the models [62].

A meta model for our proposed paradigm is provided in [41]. Over the years, we have instantiated this meta model for different domains of applications where the resulting artefact models have been evaluated and disseminated to practice. A discussion of those models is provided in Sect. 3. The models had all different contents, but they all relied on the same notion of artefact orientation that we introduce in the following.

2.2 Fundamentals and Terminology used in Artefact Orientation

In the following, we introduce the basic concepts and the terminology used for artefact-based RE as it results from our previous work [41, 42] and as it shall be used in context of this article. The most important terms are listed in Table 1.

Each artefact captures two views: A *structure* view and a *content* view. The structure view captures for each artefact type (e.g., requirements specification) the content items to be considered (e.g., use case model). For each content item, we define the content view via the modelling concepts, e.g., the elements and (content) relations of a use case model and different description techniques that can be used to instantiate these concepts and form the *representation* of an artefact. The structure model is used to couple the contents to the elements

Table 1: Terminology used in this article.

Term	Description
Project	Software development effort aimed at the construction of a (software) system through the application (execution) of a development process model (see also [23]).
Requirements Engineering reference model	Standardised organisational blueprint that includes the description of the generic process (definition) to follow, the artefacts to be generated, as well as roles involved (see also [23, 30]).
Process	A process is a series of actions that produce something or that lead to a particular result [46].
Artefact	Deliverable of major interest that abstracts from contents of a specification document. It is used as input, output, or as an intermediate result of a process step (see also [41]).
Artefact model	Model that defines a family of artefacts and their dependencies.
Method	An information systems development method is likely to include a series of phases with subphases, each having expected outputs (or artefacts); a series of techniques; a series of tools; a training scheme and some underlying philosophy [1, p. 44].

necessary to define a process, i.e., to roles, methods, and milestones. Regarding the methods and description techniques for creating the contents (e.g. UML or natural text), we leave open which one to choose, as long as the contents and relationships proposed by the artefact model are specified.

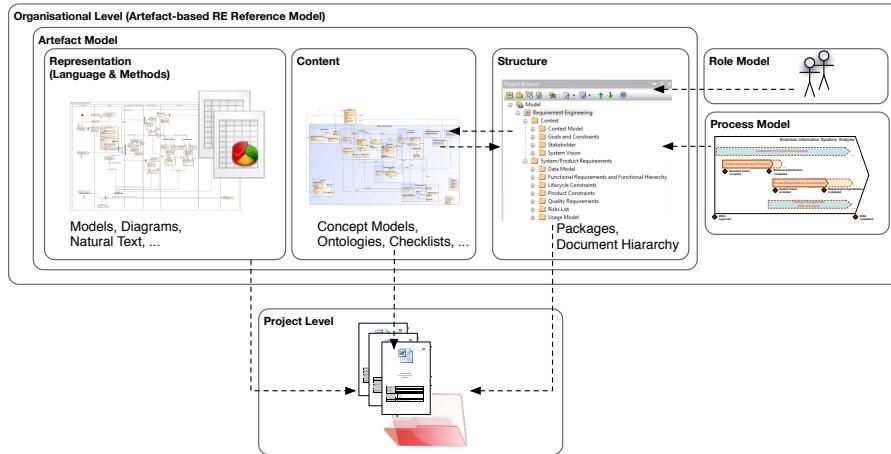


Fig. 2: Principles of artefact orientation.

Same as for activity-oriented approaches, we consider a guiding backbone necessary for artefact-based approaches, which is constituted by the artefact models (see Fig. 2, left side). However, instead of defining the artefact-based

requirements engineering approach on the basis of interconnected phases, activities and methods, we define the approach on the basis of the artefacts and their dependencies. We define roles and responsibilities for the artefacts to be created as well as the milestones, which define until when to complete, quality assure, and deliver an artefact. This reference model at the organisational level thereby allows to flexibly guide RE at the project level as the way of creating the artefacts is left open to the project participants.

3 Our Artefact-based RE Approaches and their Synthesis

The backgrounds of AMDiRE are various fundamental and conceptual approaches from our previous research. After evaluating and disseminating those approaches into practice during the past 6 years, we synthesise those experiences in the AMDiRE approach. Figure 3 illustrates an overview of our previously developed approaches.

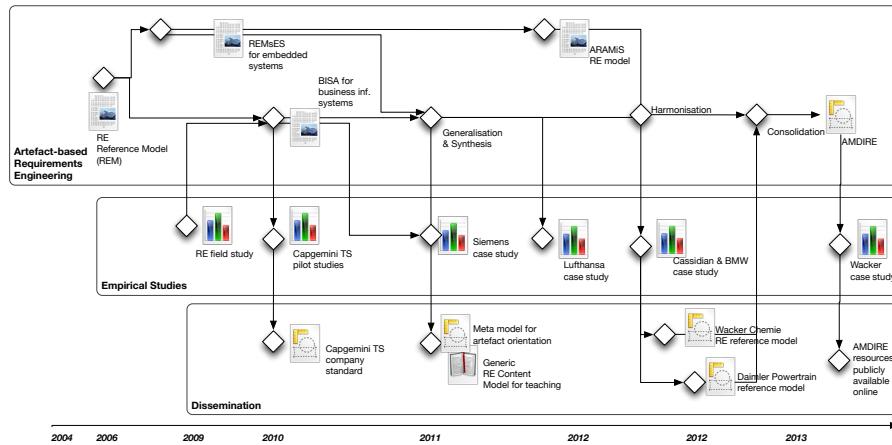


Fig. 3: Background: Development of artefact-based RE approaches.

The figure is organised into 3 layers. The upper layer shows the developed approaches to RE, followed by the second layer that illustrates major empirical evaluations of those approaches. The positive and negative results gathered from those evaluations served to steer the subsequent development. Finally, the third layer illustrates the dissemination of results (and intermediate results) into academia and practice.

In the following, we briefly introduce the development, evaluation, and dissemination of our artefact-based RE approaches, before summarising those that serve as a basis for the AMDiRE approach.

3.1 Overview of Development of Artefact-based RE Approaches at TUM

Before devoting our research to RE, we, as a research group, investigated the paradigm of artefact orientation in the area of software process models starting from 2004, as depicted in Fig. 3. In 2006, we first transferred the basic concepts of artefact orientation to RE. This effort resulted in our first reference model for artefact-based requirements engineering: the requirements engineering reference model (REM) [21, 63]. REM resulted from a research co-operation between the Technische Universität München (TUM) and Siemens Corporate Research. The model defines the structure of goals, requirements and specifications within a proposed taxonomy-based guideline and informally describes dependencies between the elements of the guideline based on proposed refinement principles. Although REM was not intended to capture details of particular application domains, the approach provided a first consolidated view on the previously existing guidelines and checklists available to RE, such as the VOLERE requirements specification templates [60], the IEEE recommended practice for software requirements specifications (IEEE std. 830-1998) [26], or more practical guidelines such as the one of Wiegers [69].

A first domain-specific artefact-based approach was developed under the REMsES project [7]¹, a research collaboration with partners from academia and industry including BOSCH and Daimler. This project resulted in an artefact model for RE in the automotive domain with a strong focus on contents necessary to specify embedded reactive systems [53]. The reference model is based on two key concepts: support for abstraction levels and coverage of three content categories. The structure supports requirements engineers in determining which type of model they should use and what kind of abstractions they should consider in a particular project.

In parallel to this development, we worked on another artefact-based RE approach for business information systems analysis (BISA) [39, 38] as part of a bilateral research co-operation between the TUM and Capgemini Technology Services, the German branch of the Capgemini group for custom software development. The resulting BISA approach is a model-based RE approach that consists of (1) an artefact abstraction model with horizontal abstraction and modelling views, (2) a concept model that defines the possible notions for producing the models, and finally (3) a method description that defines the activities and tasks of the RE process. After two years of development and evaluation in 16 pilot projects the approach became the company standard for RE.

The evaluations of both approaches showed benefits as well as shortcomings. In contrast to the REMsES approach, the BISA approach proved to better support the specification of detailed results due to the detailed concept model, but needed training and coaching. Also, the method descriptions in BISA increased the complexity unnecessarily.

¹ REMsES guide available at <http://www.remses.org>

The subsequent consolidation thus included three steps. First, we integrated the artefact-based approaches into our previously developed software process models [39] to address a broader audience and to steer further evaluations. Second, we generalised and synthesised our approaches to establish a meta model for artefact orientation. Third, we conducted additional case studies where we applied the consolidated approach in different socio-economic contexts to test the external validity.

The meta model for artefact orientation [41] unifies the different views we had so far on artefact models, including a coarse-grained view on the structure of artefact models as given in development process models and a detailed content view as given in model-based development where we detailed the topics with concrete concept models (successfully introduced by BISA). The coarse view aimed at supporting a flexible process definition, while the concept model aimed at offering guidance for the creation of detailed results. Both views result in our understanding on the constructs necessary to define an artefact, as discussed in Sect. 2.2.

The first case study of the consolidated approach was performed with a street traffic management business unit from Siemens [42]. In this case study, we empirically analysed the different benefits and shortcomings of our artefact-based RE approach, but remained aware that the empirical evidence was limited to the particular, sensitive context of our study. The case study, however, was the first one to evaluate the available paradigms to construct RE reference models in a comparative manner by directly comparing our approach with the previously used activity-based one, followed by another study at the Deutsche Lufthansa (DLH).

The results in those studies showed us that the views captured in the meta model artefact orientation were valuable to, on the one hand, define a flexible process on the basis of a coarse structure model, and, on the other hand, to guide the creation of detailed results due to the detailed content model. However, the results also indicated that the complexity in the content model implies a higher learning curve² in the application of the approach in contrast to applying activity-based approaches.

The subsequently conducted project Automotive, Railway and Avionics in Multi-core Systems (ARAMiS)³, a German publicly funded research project where 40 partners from academia and industry worked on an integrated approach for developing cyber physical systems scenarios, resulted in an artefact-based RE approach [52] with a less complex content model. Subsequent case studies at BMW and Cassidian [54] followed the same study design as defined for the study at Siemens and strengthened our confidence on the general benefits of artefact orientation, but also that the complexity in the given content model, although necessary to support a high level of detail in the results, ham-

² We understand learning curve in the sense of being related to the power law of practice, such that continued application will lead to learning, as defined and described in [58, p. 3/4].

³ <http://www.projekt-aramis.de/>

pers its easy applicability. The investigation of these phenomena is in scope of current investigations as part of a family of studies [55].

Finally, after preliminary evaluation on an Automatic Cashier System [3], the first empirical study on AMDiRE has been completed at Wacker Chemie (reported in Sec. 5.2.4), see right side of Fig. 3. Another study investigated the applicability for constructing a RE model in the context of agile methods [73]. Further studies to contribute to the family of studies [55] are currently in progress. For dissemination, a set of resources — cheat sheet, Magic Draw plugin, example specifications, and evaluation template — is available online [37].

3.2 Summary of Approaches and Their Characteristics

Table 2 summarises those artefact-based RE approaches, which serve as a basis for the AMDiRE approach. We take into account their structuring into basic components as well as their contents.

Table 2: List of approaches with their evaluations and characteristics.

Approach	Components	Characteristic	Evaluation	References
REM	Artefact model	Structure model	N/A	Model [21], Tool [63]
REMSeS	Artefact model and modelling techniques	Checklists and modelling techniques for embedded systems	Daimler, BOSCH	Model&Eval. [53], http://www.remses.org
ARAMiS	Generic content model	Domain-independent structure model for cyber-physical systems and partial concept model (for tooling)	BMW, Cassidian	Model&Eval. [52], http://www.projekt-aramis.de/
BISA	Artefact model, process elements, customisation approach,	Structure model and concept model for the purpose of process integration	CapGemini (N/A), Siemens	Model [39, 38], Evaluation (Siemens) [42]

While the first artefact-based approaches served as initial guidelines, they provided only limited guidance for the content creation as their focus was the establishment of a basic structure model and the inclusion of checklists for the content creation. The BISA approach furthermore incorporated a detailed concept model. This allowed us to support the creation of detailed results as the artefact model made explicit the concepts of an application domain. The structure model additionally supported the process integration, i.e. the coupling of the content items to milestones or roles. Other components which

turned out to be necessary for application in project environments were a customisation approach as well as tool support relying on the concept model from which we inferred UML profiles. Further information can be found in [41, 38].

3.3 Synthesis of Established Concepts

As discussed in our research method in the introduction (page 4), we synthesised the established concepts to develop the AMDiRE approach. To this end, we considered the basic components provided by BISA and the content items provided by ARAMiS, which serve as a lessons-learnt-based set of content items relevant for different application domains.

In order to ensure the applicability of AMDiRE, we made use of the process elements and the customisation approach of BISA that both rely on a structure model. This idea of a plain structure model, in turn, results from ARAMiS, and logically groups modelling concepts constituted by the BISA concept model.

As AMDiRE is intended to be broadly applicable across application domains, we aggregated, where possible and reasonable, those elements that specify same or similar concepts for different domains into one content item. For instance, AMDiRE includes an element *Domain Model* that includes business process modelling as well as the operational context with hardware and software - the first is relevant to business information systems, the latter for the domain of embedded reactive systems.

In summary, the resulting artefact-based approach shall allow for the specification of detailed results (supported by the detailed concept model) and, at the same time, be easy to use (supported by a simplified structure model). Both views and relating process elements are introduced in the following section.

4 The AMDiRE Approach

In the following, we describe the AMDiRE approach resulting from the consolidation of the fundamental, conceptual, as well as empirical contributions introduced in the previous section. We first introduce the basic components of the approach and give an overview of the artefact types, the role model and the process model, as well as further constructs used to operationalise AMDiRE in individual socio-economic contexts. We then introduce the artefact model in detail.

4.1 Artefact Types, Roles, and Milestones of AMDiRE

Figure 4 shows the basic components that build up the AMDiRE artefact-based RE approach. Those components result from our understanding of the

artefact-based paradigm as introduced in Sect. 2.2 and lessons learnt introduced in Sect. 3.

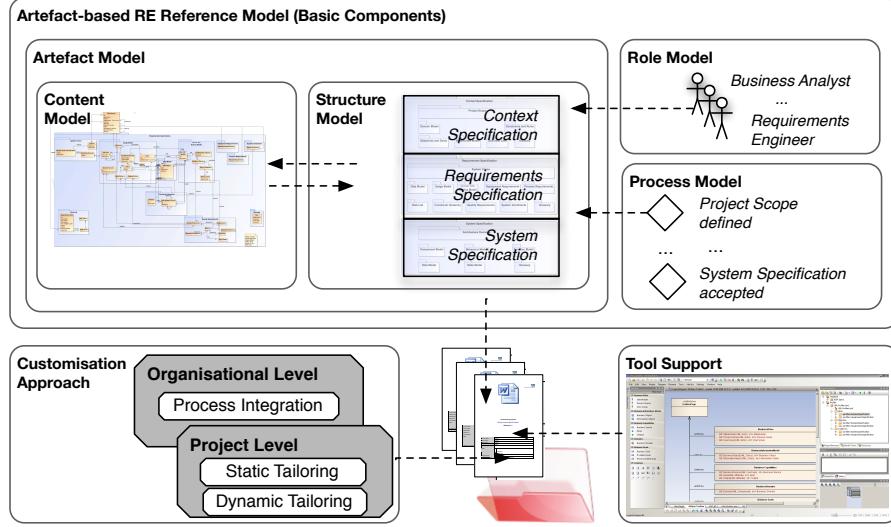


Fig. 4: Overview of the AMDiRE components.

The artefact model represents the backbone of the approach and encompasses concepts used to specify the contents of the artefacts. This model consists of two basic sub-models: the content model and the structure model. The content model abstracts from the modelling concepts used for a particular family of systems and only scopes the type of information needed. The structure model gives a logical structuring to those concepts and is used for the integration with the role model and the process model.

We distinguish in total three artefact types (Figs. 4 and 5):

1. The *context specification* defines the context of the system under consideration including a specification of the overall project scope, the stakeholders, rules, goals, and constraints as well as a specification of the domain model. The latter comprises, for example, business processes to be supported without, however, defining how the system is intended to be used in context of those processes.
2. The *requirements specification* comprises the requirements on the system under consideration taking a black-box view on the system, i.e. we specify requirements from a user's perspective without constraining the internal realisation of the system.
3. The *system specification* finally comprises a glass-box view on the internal realisation of a system including a logical component architecture and a specification of the behaviour realisation with, e.g., functions and inter-

faces. While we consider the context and the requirements specification to address the problem space, the system specification addresses the solution space and is the interface to tie in with the design phase.

Figure 5 shows the artefact types in relation to roles and responsibilities (left side) and in relation to milestones (right side) which we use to integrate the model into a process.

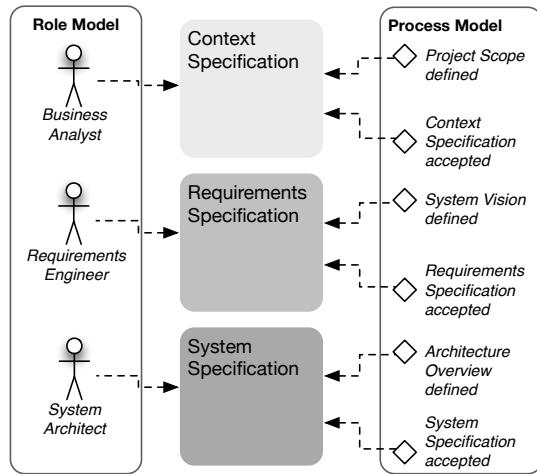


Fig. 5: Overview of artefacts types, roles, and milestones.

For each artefact type, we define one particular role, which has the responsibility for an artefact type, independent of other potentially supporting roles provided by the software process model (e.g., quality manager), and independent of whether same persons are assigned to different roles in a project.

1. The *Business Analyst* has the responsibility for the context specification and is expected to have the necessary domain knowledge, e.g. regarding the business processes, typical stakeholders, or constraints and rules.
2. The *Requirements Engineer* has the responsibility for the requirements specification and serves also as a mediator between the business analyst and the system architect.
3. The *System Architect* has the responsibility for the system specification and is expected to have technical knowledge. In dependency to the application domain, we can further distinguish between a role for the logical architecture and a role for the technical architecture (e.g. in the area of business information systems).

For each artefact type, we furthermore define two milestones. The first milestone defines the point in time in which the first content item is defined, thus, reflecting a certain maturity of the content in the artefact as the first content

items serve the purpose of a summary for subsequent contents. For instance, the system vision in the requirements specification comprises an overview of the major use cases; its definition and agreement indicate that the use cases are sufficiently defined to be further refined and modelled and, thus, allowing, for example, for first cost estimations based on function points. The second milestone of each artefact indicates the point in time when an artefact is finalised, respectively formally accepted.

Those milestones serve the purpose of a process integration and instantiation as they give us the opportunity to formally embed the artefacts into project-specific decisions. These decisions are to be taken at a specific point in time, such as when to conduct first cost estimations, when changes in the requirements should be formally defined via change requests, or when to take the contents in the specifications for a project classification and customisation (tailoring).

4.2 AMDiRE Artefact Model

In the following, we introduce the refinement principles over the three levels of abstraction by giving an overview of the content-related dependencies between the artefact types. Afterwards, we outline the content model.

The artefact model is specified using the following notational aspects of UML class diagrams:

- We denote the hierarchical structuring of the structure model with packages.
- For the definition of the content model, we use a class diagram.
- For content items that are crucial for only a specific application domain, but irrelevant for another, we use the stereotype $\langle\langle Domain \rangle\rangle$, such as business process models being crucial for the domain of business information systems, but irrelevant for the domain of embedded reactive systems.

4.2.1 Refinement Principles and Artefact Dependencies

Figure 6 organises the three artefact types in a top-down hierarchy reflected in the three previously introduced levels of abstraction (see also Fig. 5) and shows the refinement principles we use when modelling requirements and system properties. For reasons of complexity, we intentionally refrain from a complete overview of the artefact model and instead focus on selected concept types to introduce the content-related dependencies.

In the context specification, we capture behaviour in form of stakeholders performing selected processes. We specify, for example, a business process model that dictates functional behaviour by a set of process steps interrelated in a causal manner.

In the requirements specification, we select those steps to be supported by a system and specify how the system is intended to be used in interaction

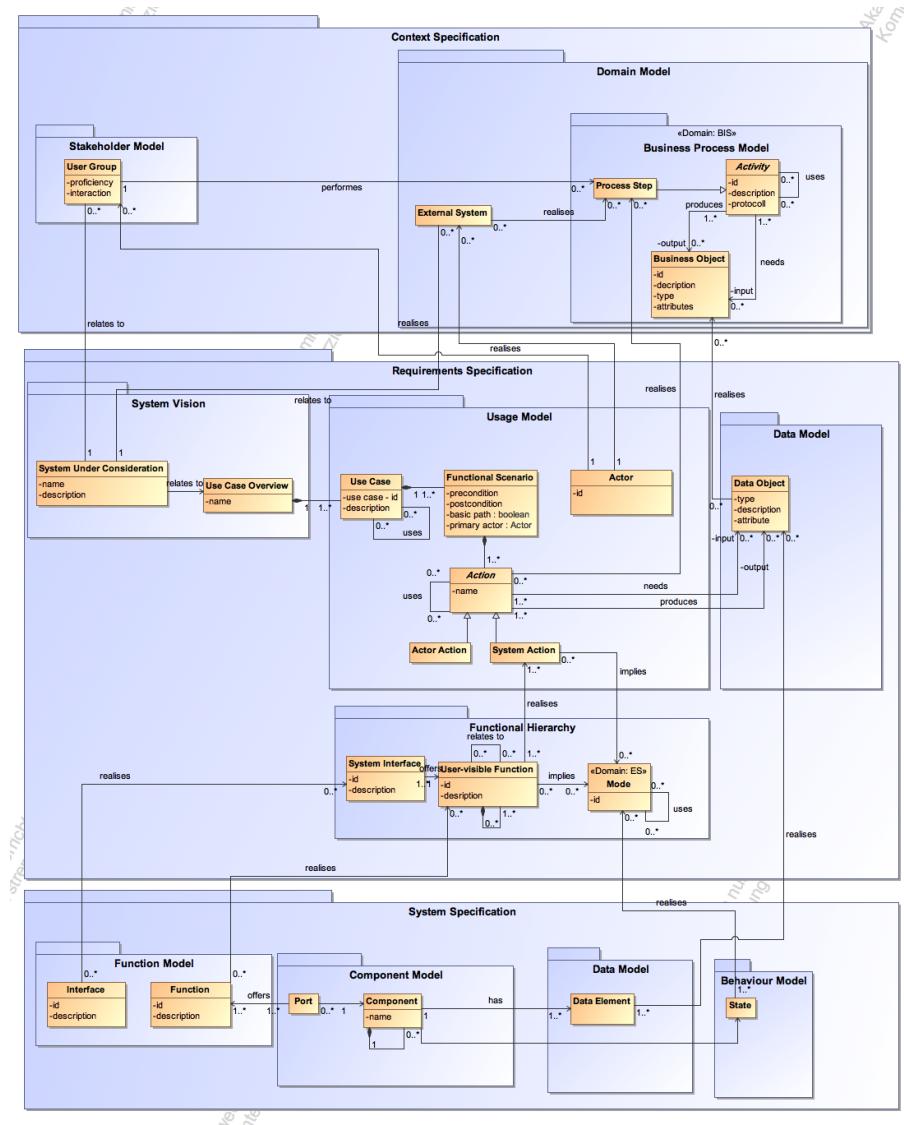


Fig. 6: Refinement and Realisation Principles in AMDiRE (complete relations are visible in the detailed subfigures in the appendix).

with the user groups. The content-related dependencies between both artefact types is given as follows:

- *Actors* to which we refer in usage models (e.g. in use case models specified via UML activity diagrams) realise either *User Groups* or *External Systems*.

- *Data Objects* to which we refer to specify which information is used as input or output to our system from a black-box perspective realise selected *Business Objects*.
- *Actions* to which we refer when specifying external system behaviour (e.g. via usage models) realise selected *Process Steps* by either defining actions an actor performs or actions a system shall automatise.

In the system specification, we then specify how the system will realise the functional external behaviour within a component architecture and its internal behaviour. To this end, we define the following content-related dependencies:

- *System Functions*, which are provided by components, realise user-visible functions, i.e. those *System Actions* in an *Usage Model* specified in the requirements specification. Same holds for the *System Interfaces*, which realise the identified (typed) *Interfaces*. The realisation dependencies are limited to the external interfaces and functions as we enrich the system specification during design activities with additional internal functions and interfaces between the components of a (logical and/or technical) component architecture.
- *Data Elements*, which are allocated to specific components and which are processed by system functions over their typed interfaces, realise the *Data Objects* specified in the requirements specification.
- *States* realise the *Modes* in a requirements specification and form system behaviour by interrelating the different states of a system via state machines. Similar as it is the case for the transition from the context to the requirements being of interest for business information systems, the transition at hand is of interest when addressing the domain of embedded reactive systems where we identify the relevant states during requirements engineering (reflected in modes).

Refinement of Quality-related Properties Further elements relevant for the transition between two levels of abstraction (not shown in Fig. 6) are those we use to specify quality and quality requirements as part of more general non-functional requirements. In our understanding, non-functional requirements cover system-related quality aspects as well as requirements on properties of the development project specified, e.g., via *Process Requirements* [22]. Quality is a multifaceted topic with different views of the term *Quality* [20, 32], and no commonly accepted definition [22]. To avoid ambiguities and to be precise with what we consider as *Quality*, we explicitly refer to a quality definition model. Due to the focus we have in RE on specifying activities with business processes and use cases, we rely on the activity-based quality definition model [16, 15] by TUM, which is based on early efforts of Boehm et al. [5] and McCall et al. [11].

The basic idea in this model is to define quality via a set of system properties and their associations to activities carried out during the use of the system [15]. For AMDiRE, this means that we define quality via (1) abstract

goals [14] over different levels of abstraction to motivate the refinement of behaviour. This behaviour is specified via (2) generic scenarios that define which non-functional activities the system shall support (e.g., activities the administrator carries out), and, finally, refine those scenarios to (3) assessable quality requirements, which then are used to motivate design decision in a component architecture. Consequently, we do not follow a strict separation of concerns regarding quality, but see the notion of quality to affect and define behavioural properties of a system and structural properties as well.

4.2.2 AMDiRE Content Model

The AMDiRE content model is structured into three artefact types that encompass over 70 elements and various relations. For the sake of clarity, we provide a simplified view on the content model while detailed information of all content items and the underlying concept model are provided in Appendix A. Figure 7 illustrates the content model and shows a sketch for each content item. For reasons of reducing complexity, we likewise depict only a subset of the dependencies between the content items (for the complete list, see Appendix A). As introduced in the previous sections, AMDiRE relies on a refinement notion for functional as well as non-functional modelling concepts.

Starting at the top, in the *Context Specification*, the *Project Scope* defines the relevant problem to be addressed by a project and the primary scope. The *Stakeholder Model* is used to capture the most relevant stakeholders and the relationships and are used as a central definition of key reporting lines and one important rationale for requirements and goals. *Goals* are specified, e.g., in a graph form, and serve as a means to steer the specification of a business process model in the domain model. The *Domain Model* provides information on the operational environment.

In the *Requirements Specification*, the *System Vision* defines the basic idea of the system under consideration and the stakeholders of a project agree on a system scope (major features and use cases) as well as its boundaries specified via a context diagram or a rich picture. To capture functional behaviour in the *Usage Model*, we define for each identified use case how future users intend to use the system in interaction. A *Service Model* is used as a complementary means to define which services the system shall offer – in contrast to a use case model not necessarily defining the relation to actors but, instead, the causal relations between the services. We further use the system-supported actions in a use case model, e.g., specified via UML activity diagrams, to select candidates for user-visible system functions, which we structure and refine in a *Functional Hierarchy*. This hierarchy builds the point of entry into the system specification.

In the *System Specification*, we finally allocate the *Functions* of a functional hierarchy to *Components*, define their syntactic interfaces and their internal *Behaviour* with, for example, automata. This behaviour specification also serves the identification of the (typed) entities defined in relation to each other in a *Data Model*.

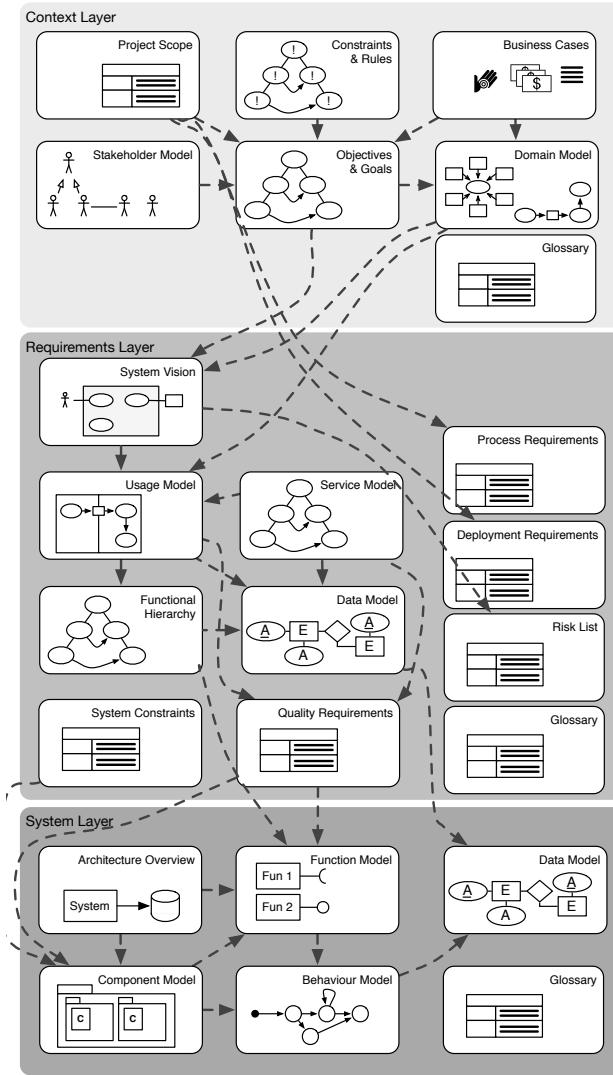


Fig. 7: Overview of AMDiRE Content Model in a simplified manner.

4.2.3 Customisation Approach and Tool Support

The operationalisation of AMDiRE in a particular socio-economic context is done by two means. In a tailoring approach, AMDiRE is customised over two levels of abstraction:

1. Organisational level: At the organisational level, we consider the company-specific customisation of AMDiRE, i.e. the process integration into an in-

dividual organisational reference software process model such as RUP (or a company-specific derivate).

2. Project level: At the project level, we consider the instantiation of AMDiRE (initial artefact creation and decision for specific content items, assignment of roles and definition of milestones), which is known as static tailoring, and the dynamic tailoring during project execution. The latter considers the situation-aware creation of content items in dependency to project-specific situation that affect the need to create particular items and ones that affect the possibility to create particular items. An exemplary set of project influences and their dependency to RE artefacts can be found in [43].

Both customisation at organisational level and at project level are introduced in [38]. In order to further operationalise AMDiRE, we rely on tool support. This is currently prototypically realised with an extension of the

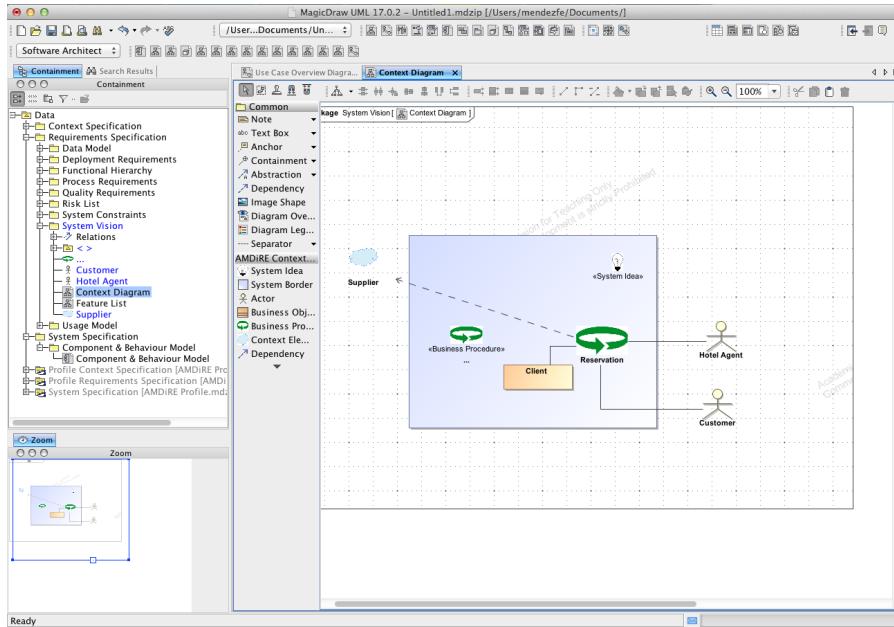


Fig. 8: Screenshot of model-based tool support at the example of a context diagram.

model-based CASE tool MagicDraw⁴ by defining an UML profile based on the content model provided by AMDiRE (see Fig. 8). The tool extension is available online [37].

⁴ <http://www.nomagic.com>

4.2.4 Example Application

This section shows an illustrative excerpt of example models for a fictitious Automatic Teller Machine (ATM). The example was developed for a Requirements Engineering lecture at Technische Universität München following the artefact model of AMDiRE and using the tool extension depicted in the previous section. The complete example models can be obtained together with the tool extension from our online resources [37].

Figure 9 shows the exemplary system vision, the stakeholder model, the goal model, and a scenario from one of the use cases created. The system

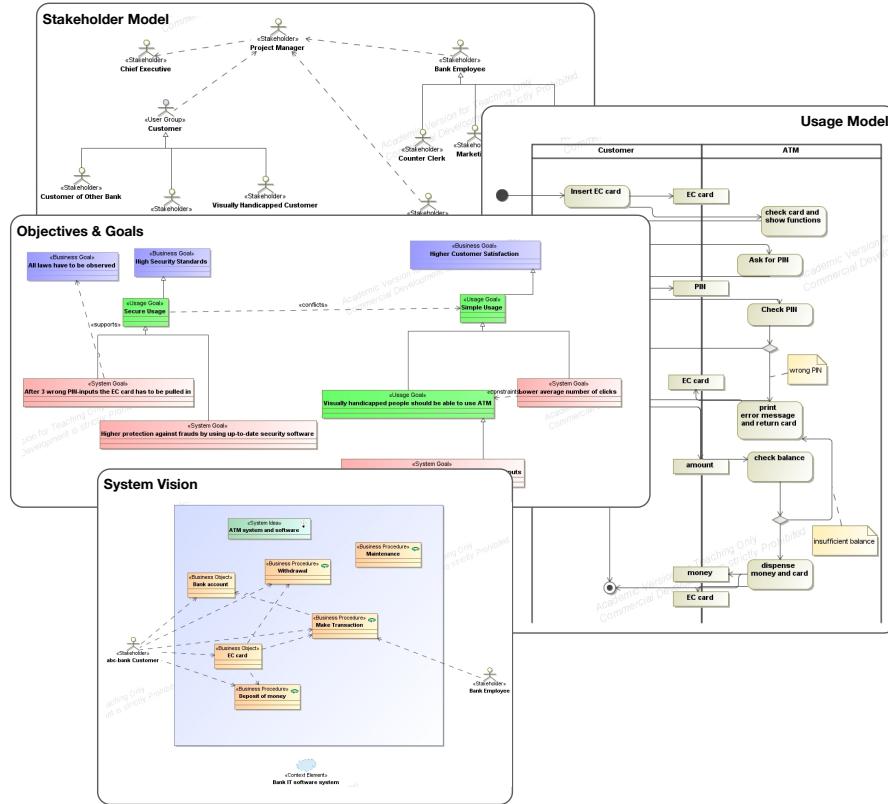


Fig. 9: Illustrative excerpts of the ATM example

vision is the agreed-on vision of the ATM that denotes the system border and the most central features, like *withdrawal* and *transaction*. The stakeholder model includes business stakeholders and user groups in various hierarchies. The goal model includes business goals like *higher customer satisfaction*, usage goals like *visually handicapped should be able to use ATM*, and system goals

like *high protection against fraud*. The activity diagram illustrates the scenario of a user *withdrawing money* from the ATM.

5 Experiences, Evaluations, and Lessons Learnt

We have developed a number of artefact models, conducted case studies with different companies, and gained different experiences. In the following, we first discuss the experiences we gained throughout the construction of artefact models. Furthermore, we discuss the industrial evaluations we performed. This shall give a picture of benefits and shortcoming in the construction and the application of the various notions in artefact-based RE. We finally conclude with a discussion of general lessons learnt in the construction and application of artefact orientation in industrially hosted environments.

5.1 Experiences in the Construction of Artefact-based RE Approaches

Over the last years, we have performed a series of different research cooperations in the field of artefact-based RE improvement where we developed company-specific artefact-based RE approaches.

The research approach that has proven feasible for us throughout a number of projects is to follow a problem-driven exploration where we apply concepts of empirical design science [70, 24] to build a company-specific RE approach as part of an RE improvement endeavour. Figure 10 illustrates this research approach, separating two basic phases: The actual problem investigation, and the design and validation phase.

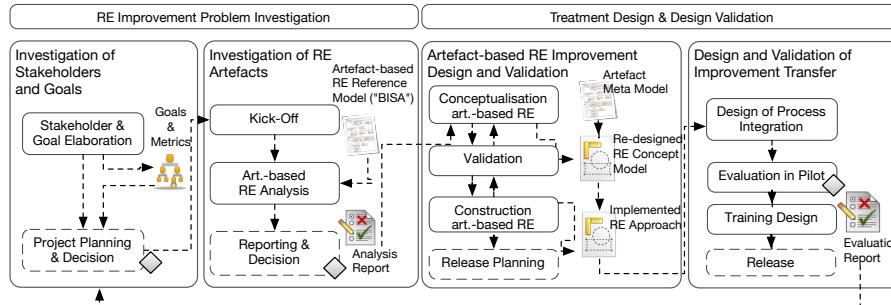


Fig. 10: Procedure for constructing artefact-based RE approaches.

Briefly summarised, we begin with a kickoff where the most important stakeholders for the artefact model and their roles are identified, followed by semi-structured interviews with these. Subsequently, we review requirements

documents that have been produced by the company according to their current reference model and/or practice and compare these to our content model (gap analysis). The differences found in that comparison are discussed with a representative set of stakeholders in a workshop where we ask for reasons for specific elements we had expected but were not being documented and for extra elements that were documented but not present in our concept model. This allows for a first evaluation where our reference model should deviate for the later elaboration of the to-be model.

On this basis, we develop an artefact model that is tailored to the needs of the company by performing a series of action research workshops, and we perform a pilot study and/or a comparative case study to evaluate whether the artefact model brings the desired improvement for their requirements engineering practice. A detailed explanation of our approach to construct artefact models is provided in [45].

Apart from the methodological experiences we made when building up various artefact models (introduced in [45]), for example, regarding the use of action research workshops to support knowledge transfer [71], we gained experience in structuring artefact models at a more syntactic layer.

In [41], we already discussed first experiences regarding the notion of artefact orientation and inferred a meta model for the paradigm. We discussed that we need to consider different views on artefact models depending on company-specific objectives. Those objectives affect the way artefact models are structured (e.g. by defining a detailed concept model via a data model or by defining the content model via a checklist). The models can have different levels of detail according to how much detail the company wants in the model, leading to more complexity, which affects ease of use. Consequently, an increased level of detail requires a higher learning curve by the applying company. This trade-off between higher level of detail and ease of use has to be addressed.

In Table 2 (Sect. 3.2), we selected previously developed artefact-based RE approaches and their characteristics. In Table 3, we structure different approaches in a similar way and summarise the main objectives of the company and their effects on the establishment of the artefact models, as well as the effort spent in their development (in person months).

Top to bottom in Table 3, we can see

- An increase in the details the artefact model incorporates by its underlying content models. Those range from detailed data models in case of the BISA approach to support seamless modelling to artefact models that incorporate checklists and selected concepts to clarify the notion of selected contents (e.g., use cases).
- A decrease of the effort spent in the construction of those models. Taking into account our own learning curve during the construction of those models, we still have efforts that differ by a factor of 18. Apart from light-weight versus heavy-weight models, this is mainly due to the up-front budgeting and time framing for the projects.

Table 3: Artefact models, their objective and resulting characteristics.

Approach	Objectives	Characteristics & Effort
#A BISA / Quasar Requirements (Capgemini TS)	Process integration and seamless modelling	Structure model and content model defined via a data model and conformance constraints, 112 PM
#B Generic Content Model (ARAMiS)	Common terminology across the 40 project partners and a concept model for tool support and guidance	Structure model and implemented concept model / UML profile, 20 PM
#C Artefact Model for Wacker Chemie	Template-oriented checklist to integrate RE artefacts into quality assurance	Structure model and templates, detailed concept model for use cases and the relation to test cases, and process elements, 8 PM
#D Powertrain Artefact Model (Daimler)	Clarifying the notion of requirements contents	Structure model and detailed concept model without roles, milestones or other process elements, 6 PM

The simpler models developed in context of Daimler (# D) and Wacker Chemie (#C) follow a similar objective where we established the artefact-based RE approach to clarify the basic RE concepts and corresponding terminology. Those models serve the purpose of giving a quick overview of elementary modelling concepts such as use case models, their allocation to requirements-specific artefacts and their dependencies to surrounding development phases. At Wacker, the process integration could explain the additional 2 PM. During this integration, we defined roles and responsibilities, a quality assurance process and the integration of a tailoring profile (see Sect. 4.2.3). The effects of following those objectives can be observed, however, especially in the way the artefact models were created. In scope were the structure models to define which content items were of general interest. For each content item, we provided a brief guidance in form of a checklist and only specified the concept model in detail where necessary (e.g. for the use case model to define the dependency to testing).

In contrast, the objectives followed by the two more detailed models (#A and #B) were mainly to support seamless modelling and tool support. The implication for the artefact model was that the content models were specified in full via a data model to define which concepts and relations have to be considered during the artefact creation. Both concept models were also enriched by a structure model to ease the process integration.

The effort in the detailed models mostly arose from:

1. Terminological and conceptual discussions at workshops, frequent action research workshops to realise the concepts, and a longer review and release process at the partners' sites as the approaches had not anymore the character of being a checklist.

2. The development of training and coaching material as well as tool support.
3. The supervision of pilot projects and coaching during those pilot studies.
4. More general, a missing willingness to organisational change we could observe in the projects as project participants were confronted with complex models and needed a longer learning curve to understand the basic concepts and tailoring mechanisms.

5.2 Evaluations: Results from Industrial Case Studies

We performed a series of evaluations with the previously developed artefact models on which AMDiRE relies. We complement this evaluation with a case study where we directly used AMDiRE as a reference model to establish a company-specific artefact-based RE approach following the principles described in the previous Sect. 5.1.

5.2.1 Summary of Previously Conducted Comparative Case Studies

We have performed three comparative, industrial case studies that we briefly discuss with regard to their results. These studies each evaluated an artefact-based approach in direct comparison to the formerly used activity-based approach and were performed with partners at Siemens, BMW, and Cassidian. Table 4 provides the type of studies and their major results.

In the following, we summarise those previously published case study designs and results relevant to the context of the article while details on the study designs and the full results can be taken from references listed in Table 4.

Table 4: Comparative industrial case studies.

Study	Approach	Partner	Results
Traffic Management	Lights System	#A (2011)	Siemens Improvement of, inter alia, syntactic consistency and completeness, ease of use, effectivity, structuredness and ease perception [42]
Intelligent Information Systems (IIS)	Info- tainment Systems	#B (2012)	BMW Improvement of effectivity, productivity, and adequacy [54, 52]
Unmanned Aircraft System (UAS)	Air- craft System	#B (2012)	Cassidian Improvement of ease of use, effectivity, unambiguousness and adequacy [54, 52]

5.2.2 Case Study Designs Overview

As shown in Table 3 and in Sect. 3 introducing the previously developed models, we initiated our case studies with smaller studies at Capgemini TS

where we evaluated the approach in 12 internal pilot projects. As those studies remain unpublished due to a non-disclosure agreement, and because we aimed at increasing the external validity, we applied our approach #A in nother socio-economic contexts. After this evaluation, we continued the development of our artefact models leading to approach #B which, again, we evaluated to test its sensitivity in a socio-economic context.

Overall, we conducted three case studies where we evaluated two approaches (see Table 4). We evaluated approach #A at Siemens and approach #B at Cassidian and at BMW. The objectives of the case studies were defined, at the time of conducting the studies, according to the improvement goals of the projects. That is, in dependency to the defined goals, we formulated the research questions and the evaluation criteria. However, the overall study design remained the same for all studies to allow for a comparability of the results.

In each study, we conducted a series of workshops with our industry partners and specified the requirements for a selected subsystem following the established artefact-based RE approach. We then conducted an assessment workshop with interviews where the subject evaluated the created RE artefacts as well as the way of working in direct comparison to the artefacts and the process dictated by the (activity-based) RE reference model previously used in the same context. For the assessment, we intentionally conducted one interview session with all involved participants to enable joint discussions among the participants and to get a final agreement on the ratings (without checking the inter-rater consistency). In all case studies, we involved a requirements engineer (in case of Siemens represented by the role of a product manager) and the overall project lead responsible for broader project management activities.

The rating was conducted by answering a questionnaire with a series of open and closed questions. The closed questions aimed at rating single criterion, such as *ease of use* on a Likert Scale while the open questions served to justify the ratings. The choice of the criteria was performed in discussion with the project participants according to the (improvement) goals and can be taken from the Kiviat diagrams in subsequent result sections.

5.2.3 Case Study Results Overview

In the following, we summarise the results from the previously conducted case studies relevant to the context of this article.

Figure 11(a) shows the results from the internal rating at Siemens. We complemented the internal rating with an external one where we called in a neutral person not involved in the RE workshops to support a more unbiased evaluation. The results of the external rating are shown in Fig. 11(b). Both evaluations consider the process when applying the artefact-based RE approach as well as the quality in the created artefacts. Although some criteria had a low consistency between the internal and external rating (e.g., the traceability), the application of the detailed artefact model resulted in general in a better support of creating syntactically consistent and testable results in

direct comparison to the previously used RE reference model. The ease of use, however, was rated worse. Details on the full results can be taken from [42].

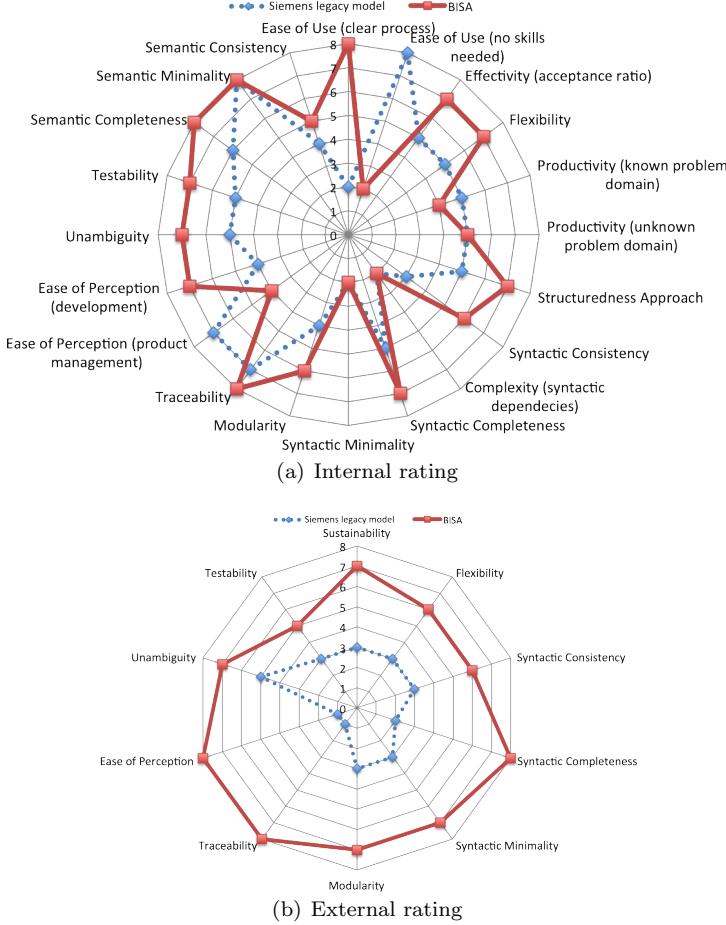


Fig. 11: Results from the case study with Siemens.

The replication studies at Cassidian and BMW revealed that the detailed concept model used to establish the tool infrastructure (see Table 3) again supported the syntactic consistency and completeness in the created artefacts. For both case study systems, we received positive feedback on the artefact-based approach and the evaluation results showed, in contrast to the study at Siemens, improvement in various aspects in the process for creating the artefacts, most importantly, the effectivity and ease of use. Our explanation for the different rating is that the model developed at Capgemini TS has

a far more comprehensive concept model capturing all particularities of the appellation domain, which, in turn, implies a stronger learning curve.

An overview of the rating of the closed questions from our automotive partner BMW is depicted in Fig. 12(a) and 12(b) and from our avionics partner Cassidian in Fig. 12(c) and 12(d). For reasons of illustration, the centre of the Kiviat diagram is labelled again with the value of zero instead of one, since otherwise the data points would overlap in the center.

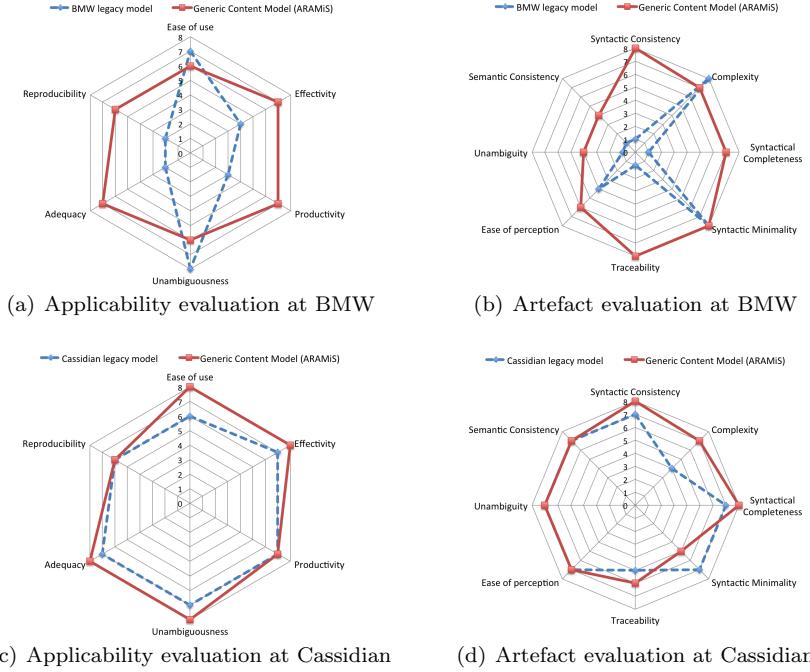


Fig. 12: Results from the ARAMiS case studies with BMW and Cassidian.

Both partners seem to perceive the ARAMiS artefact model as improvement in comparison to their previous reference model. The business units at both companies have also incorporated the artefact model into their standard. On first glance, the rating was more critical for both their own and the ARAMiS model by the automotive partner. When analysing the answers to the open questions (the rationale for their ratings), we found that some of the reasons given for a specific rating were quite similar, but rated rather differently. For example, for traceability, both stated that their own reference model provided the possibility to link requirements, but did not provide the possibility to document a rationale. At the same time, their rating of 1 at BMW versus the higher rating of 5 at Cassidian for the criterion *Traceability* shows

a considerable difference in perceiving the importance of an artefact-based reference model providing the possibility to document a rationale.

Details on the design and the results of the case studies conducted at Cassidian and BMW can be taken from [54, 52].

5.2.4 AMDiRE Case Study

Apart from the previously conducted and published case studies where we evaluated the models on which AMDiRE relies, we conducted one industrial study where we directly made use of AMDiRE. In contrast to the previously conducted comparative case studies, however, we used AMDiRE as a reference model to build a company-specific artefact model following the procedure illustrated in Sect. 5.1 and published in detail in [45].

The case study took place at Wacker Chemie, a company working in the chemical business with quarters in Munich. The department with which we worked focuses on the engineering of development processes for standard as well as custom software development in the company-specific operation processes and their production sites. The goal during the development of the artefact-based RE reference model was to support traceability and testability of the RE artefacts. After a series of analysis workshops and workshops to construct the artefact model (see [45]), we prepare the evaluation in three pilot projects, which considered internal developments in the company. For reasons of confidentiality, we omit details of the projects and the involved subjects and remain on an abstract description. As a preparation of the evaluation in the pilot projects, we created a short presentation as internal training material and document templates for the new artefacts to be applied in the pilot projects.

The projects covered both standard software development for SAP and custom software development. As project participants served in total 8 employees of which for each project one project lead and one developer were complemented by two process engineers involved in the development of the artefact-based RE reference model. Those latter two process engineers served as coaches to train in a one-day workshop the people for applying the new artefact-based RE reference model developed on basis of AMDiRE. The study design was the same as the one from previously conducted case studies. That is, we conducted workshops to create requirements specifications following the artefact-based RE approach and concluded with an assessment where the project participants rated the new RE approach in direct comparison to the approach previously used in same environment (for previous releases). For the assessment, the project participants were provided again a questionnaire. This questionnaire was similar to the one we used in the previous studies, with minor modifications in wording. For each criteria we used in the rating, we asked a closed question where the participants should rate their agreement to a statement on a Likert-scale from 0 to 6 followed by an open question where the participant could give a rationale for their rating. Table 5 gives a condensed view on the closed questions.

Table 5: Questionnaire for the Assessment (condensed)

Criteria	Statement
Flexibility	The RE reference model allows for flexibility.
Ease of use	The RE reference model is easy to understand.
Effectivity	The RE reference model leads to the desired results.
Efficiency	I perceived the efficiency in the process as high.
Customisation	The RE reference model is tailorable according to project-specific situations of the company.
Process Integration	The RE reference model is integrated into further development activities (e.g., testing) and within the line organisation.
Structuredness Artefacts	The specification documents are well-suited to be understood respecting their structure by people not being involved into the elaboration of the specifications
Syntactic Artefact Quality	The RE reference model supports a high syntactic quality in the created RE artefacts w.r.t. consistency and completeness.
Traceability in Artefact	The RE reference model supports traceability within RE (e.g., through rationales) and between RE and further disciplines.
Semantic Artefact Quality	The RE reference model supports semantically consistent and complete results.
Testability of Artefacts	The RE reference model supports testable RE artefacts.

In the following, we summarise the case study results followed by informal feedback given during the evaluation.

Case Study Results Figure 13 summarises the results from the case study and shows with a solid red line the rating given to the artefact-based RE reference model and with a dashed blue line the rating given for the previously used reference model where the RE artefacts were underrepresented.

The goals of improving the support of traceability and testability were reached and the participants positively rated also other aspects in the artefacts created according to the new reference model. For instance, they saw an increased structuredness in the RE artefacts and rated the process integration to be better than in the previous model as the artefact-based nature clearly defined, for example, roles and responsibilities.

However, the ease of use was slightly worse than in the previous model, where the feedback indicated the higher level of detail in the reference model to be the main reason. Further feedback that supported this was that participants needed more detailed training as well as comprehensive tool support (going beyond the document templates) to better support the learning curve in applying the new reference model.

Discussion Our case study at Wacker supports our claimed general benefits of the artefact-based approach. One remarkable feedback provided by the project participants was, however, the strong need to support the application of the model with tool environments, and detailed training material as, otherwise, the ease of use in the model is strongly hampered. This observation is in tune

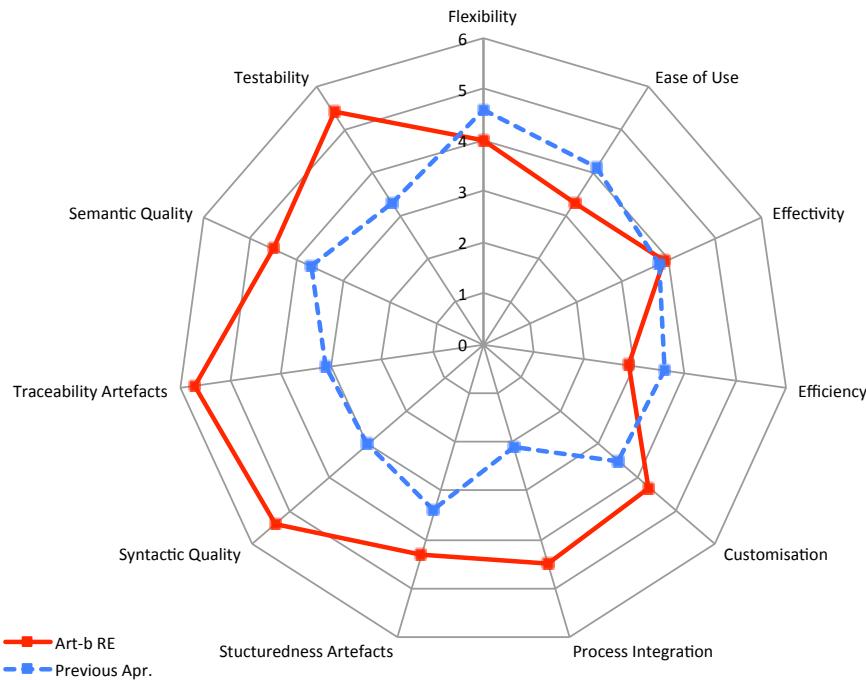


Fig. 13: Results from case study at Wacker.

with the previously conducted case studies where the ARAMiS model resulted to be better rated than the BISA model due to:

1. providing a less complex structure model that does not focus too much on particularities of single application domains, while
2. hiding the remaining complexity of the underlying concept model with its a directly applicable tool environment.

Although the AMDiRE approach covers both characteristics, we have now reached the point where we need to conduct external evaluations by unbiased researchers and practitioners not involved in the development of AMDiRE (see also the research method in the introduction on page 4). This will also show to what extent AMDiRE can be used by others and what exact effects this will have on aspects negatively rated at Wacker (relating to the learning curve).

To this end, we make publicly accessible the tools, the models, and the examples as well as the evaluation templates related to AMDiRE [37] and encourage researchers and practitioners to critically discuss and evaluate our approach in external contexts.

The article at hand thus builds the first step in the dissemination of our research results not only into practical environments with isolated problems and goals, but also back into the research community.

5.3 Lessons Learnt

Throughout the past six years of collaborating with industrial partners on improving their requirements engineering practices, we have collected a number of lessons learnt, which we summarise in the following.

Tradeoff between high level of detail and usability. Regarding more general lessons learnt in the field of artefact orientation, we have observed more detailed artefact models to increase the quality of the results as they can give detailed guidance on the concepts to use when specifying the contents. At the same time, however, those more detailed models constrain the ease of use as they implicate a higher learning curve. Obviously, simpler artefact models have the opposite effect. Whereas they are easy to apply and understand, they cannot give detailed guidance on creating detailed, syntactically consistent contents. The only mitigation we see to this problem so far is to explicitly point out this tradeoff as early as possible and in case of focussing on detailed concept models providing tool support right from the beginning.

Terminology. In every workshop we have held, the longest time was usually spent on terminology discussions. Either there were different terms available and the discussion was which one of them should be given preference, for example *context* versus *environment*. Or there was one term but different stakeholders had different interpretations of the concept it represented, for example *function*. These discussions are time-consuming but unavoidable and crucial to make sure the artefact model will be accepted by all stakeholders later on. The mitigation strategy we followed was to explicitly reserve a time slot early on to establish a list of most important terms and definitions that have to be agreed on.

Maintainability. Furthermore, we noticed the maintenance of established artefact models to be challenging. Once detailed models are established, it is difficult to integrate new concepts due to the complex content-related dependencies. We are not aware of any real mitigation to this point, but we suggest that every single modification of the model should be justified, carefully planned, well documented, and explicitly evaluated in an own pilot project to detect potential inconsistencies arising from the modification in the model as early as possible.

Incorporate down-to-earth informality. The biggest lesson learnt, although not surprising in itself but only in its extent, is the down-to-earth informality that is residing in the daily business of software systems development. This is in contrast to the high quality requirements and constraints that have to be adhered to by the final products. There is a high demand for easy to use approaches. The more details a concept model has, the more should be invested in pilot studies, coaching material, and guidelines (reflected, e.g., in the effort spent for the development of the approach at Capgemini TS).

Support learning curves. One of the quality characteristics that has always been rated worse for the new artefact-based approaches compared to the previously residing approaches is *efficiency*. Our assumption is that this is not only due to the higher simplicity of the previous models but also that the previous approaches are what the developers are accustomed to. Consequently, anything that is different from what they are used to will initially require a slightly higher effort and might therefore easily leave the impression of decreased efficiency. As a matter of fact, this strongly relates to a learning curve that is always implied by new approaches [65]. We encountered two means to support learning curves: continuous pilot studies and training and coaching. The first includes a smooth integration of developed approaches via pilot phases, which we use to get additional feedback from the project environments, and extends to long-term studies to investigate the benefits of improvement endeavours w.r.t. to project quality. The latter includes the preparation of guidance, which we believe to build an essential building block for not only introducing a new approach but for establishing it in a long run taking into account the different organisational cultures. Unfortunately, the writing of a handbook or the development of course material and trainings is often neglected. In that case the developers are likely to be resistant against innovations.

Tackle distrust right from the beginning. In projects, we have often encountered a certain distrust against new approaches. The willingness to change is rather low and a commonly known problem [40], i.e., the willingness to change results from the initial effort required to even try a different way of working, but also from general beliefs, experiences, expectations, emotions and desires. There is no universal silver-bullet to tackle this problem as, more generally speaking, the success of any improvement endeavour also depends on social skills [61, chp. 10] and eventually on politics [47]. We experienced, however, the deep involvement of different stakeholders (especially from project settings) right from the beginning of an improvement project and an honest communication are the most important aspects to tackle distrust (see also [45]).

6 Conclusion

This article described the fundamentals in artefact-based requirements engineering and presented the AMDiRE approach that emerged from six years of experience with developing artefact models and integrating them into the surrounding requirements engineering processes of various socio-economic contexts. Furthermore, we presented our evaluations in different industrial case studies and their replications in different companies as well as lessons learnt from these collaborations.

A subset of our case studies were performed as comparative studies that evaluate the application of the artefact-based RE approaches in direct comparison to the activity-based approaches previously used in the same environ-

ment. We could show how artefact orientation supports for detailed, consistent results while supporting the flexibility in the process which is especially important to RE. We also discussed, however, that we continuously need to make a trade-off between a detailed content model to support high quality results, and the resulting higher learning curve that affects the ease of use of artefact-based RE.

Our work has been incorporated into the daily requirements engineering practices of the companies with whom we collaborated. We strongly believe that this approach of carrying research into practice, adapting it to the business context, and establishing it in collaboration with the practitioners is a sustainable way of promoting the use of current research results. We hope to carry this work further into the requirements engineering community to encourage the application of artefact-based approaches and to give an example of an artefact model that can be used in a variety of application domains.

Researchers can already build their fundamental, educational, and evidence-based work upon our contributions. Our empirical results already give the opportunity to steer their requirements engineering research in a problem-driven manner. Practitioners can furthermore directly apply our model in their own socio-economic contexts with the awareness of the benefits and shortcomings of the incorporated concepts. We thus laid the first fundamental, conceptual, and empirical basis for artefact-based RE research. However, we do not claim that our approach suits every situation and that our empirical results allow for perpetually valid generalisations as we always focused with our action research case studies on specific socio-economic contexts. In fact, the research area needs further investigation and the case studies need further independent replications. The reason for independent replications is that if we go to another company and help them improve their requirements engineering using our model (which is the only reason for a company to hire us), this probably biases the results towards our model as companies get free consulting during the process. Therefore, we need an external third party to perform the validation. Consequently, we encourage researchers and practitioners to critically discuss our approach and to join us in the empirical evaluations of artefact-based RE.

Future Work. We are encouraging further replications of our comparative studies in different application domains and development contexts. Another objective for future work is to further facilitate the dissemination and empirical evaluation of our approach. We have developed a first prototype for tool support as a profile with a respective template in MagicDraw that is freely available [37]. We use our tools as an additional means for the dissemination and for continuous evaluations in pilot projects to gather more information and user feedback while preparing the development of a more elaborated stand-alone tool.

A AMDiRE Content Model

The following appendix defines the content model of AMDiRE in detail giving for each content item a definition of the used concepts.

The artefact model is specified using the following notational aspects of UML class diagrams:

- We denote the hierarchical structuring of the structure model with packages.
- For the definition of the content model, we use a class diagram.
- For content items that are crucial for only a specific application domain, but irrelevant for another, we use the stereotype $\langle\langle Domain \rangle\rangle$, such as business process models being crucial for the domain of business information systems, but irrelevant for the domain of embedded reactive systems.

A.1 Context Specification

The context specification is depicted in Fig. 14. It contains the Project Scope, the Constraints and Rules, the Stakeholder Model, the Business Case, the Objectives and Goals, the Domain Model, and the Glossary.

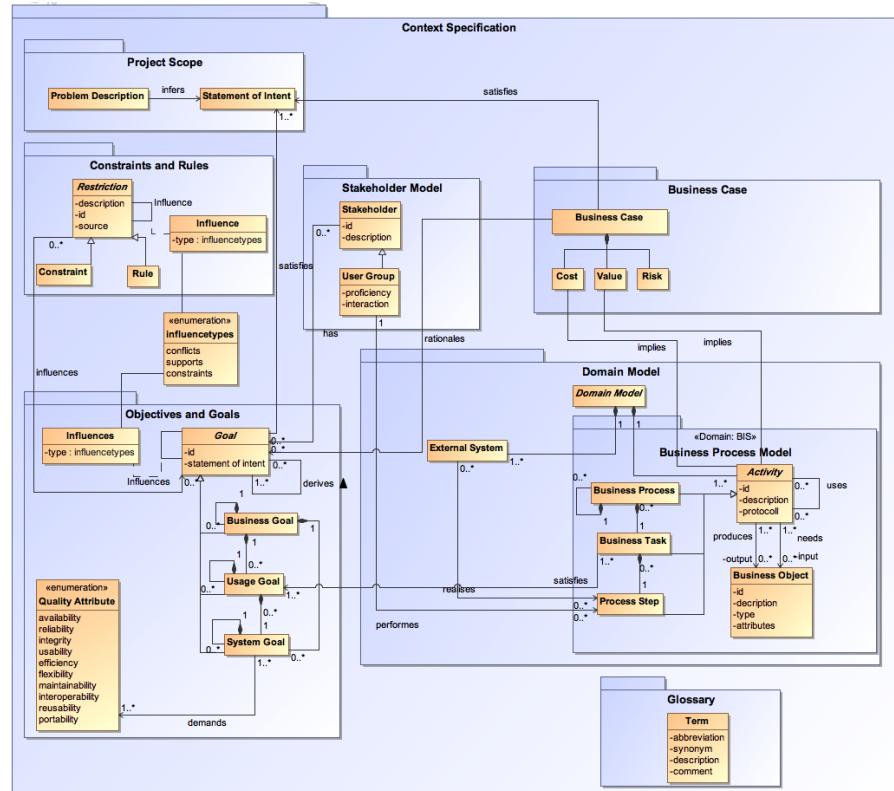


Fig. 14: The AMDiRE context specification.

A description of the content items is provided in Table 6.

Table 6: Content items in the Context Specification.

Content Item	Exempl. Notation	Description
Project Scope	Natural text	Content item of the <i>Context Specification</i> that consists of a <i>Problem Description</i> and a <i>Statement of Intent</i> , i.e. a conclusion of the objectives of a potentially resulting project.
Constraints and Rules	Natural text, graphs	Restrictions that can influence each other either in form of support or a conflict. We distinguish <i>Constraints</i> as not negotiable restrictions in the domain and <i>Rule</i> , often referred as conditional standard procedures [27]
Stakeholder Model	UML actor hierarchy, tables	<i>Stakeholders</i> comprehend individuals, groups, or institutions having the responsibility for requirements and a major interest in the project [13], while <i>User Groups</i> are a specialisation of stakeholders with a particular proficiency and involvement in interaction with the system [31].
Business Case	Natural text	Described and detailed using the additional elements of Cost, Value, and Risk. The Business Case satisfies the Statement of Intent from the Project Scope and rationalises the goals in the content item Objectives and Goals
Objectives and Goals	Goal graphs (e.g. KAOS)	Each Goal, whether it is a <i>Business Goal</i> , a <i>Usage Goal</i> , or a <i>System Goal</i> , is issued by a <i>Stakeholder</i> . Goals satisfy the <i>Statement of Intent</i> [35], they build a hierarchy, and they can influence each other in terms of conflicts, constraints, or support. Each usage goal is related to a business goal and each system goal to a usage goal. Furthermore, system goals demand one or more <i>Quality Attributes</i> [4].
Domain Model	UML activity diagrams or BPMN	The External Systems, that interact with the system under development, compose the <i>Domain Model</i> . For business information systems, the domain model is extended with a <i>Business Process Model</i> represented in various types of <i>Activities</i> that need and produce <i>Business Objects</i> . A business process model is a collection of all instances of the activities and their (causal) relations, performed by roles in order to produce some outcome of value [36]. The activities can be defined as an abstract <i>Business Process</i> , a <i>Business Task</i> (business use case), or an atomic <i>Process Step</i> – the latter represents single atomic steps performed by a <i>User Group</i> .
Glossary	Structured text	This content item contains all important <i>Terms</i> for the system under consideration and the respective project management, including their abbreviation, synonyms, and description. It shows up as well in Requirements Specification and the System Specification as more Terms are added over the course of the project.

A.2 Requirements Specification

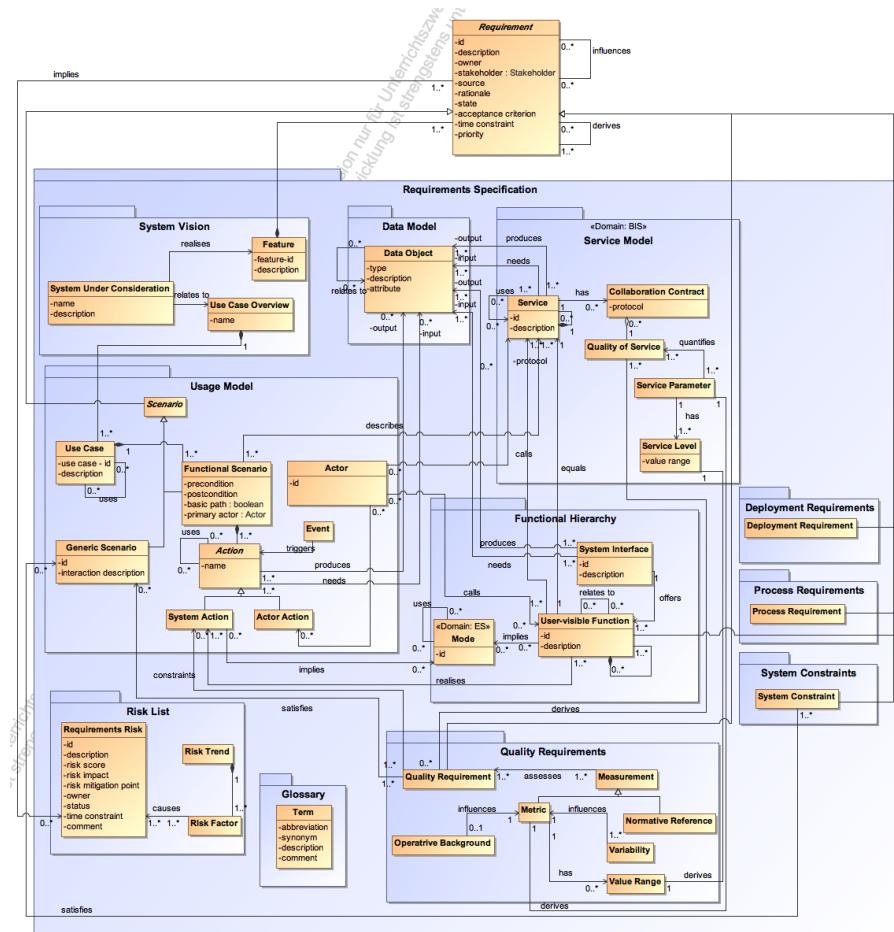


Fig. 15: The AMDiRE requirements specification.

The requirements specification is depicted in Fig. 15. It contains the System Vision, the Usage Model, the Data Model, the Service Model, the Functional Hierarchy, the Quality Requirements, the Deployment Requirements, The System Constraints, the Process Requirements, and the Risk List. A description of the content items is provided in Table 7.

Table 7: Content items in the Requirements Specification.

Content Item	Exempl. Notation	Description
System Vision	Rich picture	The system vision comprehends the system context of the <i>System Under Consideration</i> , which is intended to realise a number of <i>Features</i> . A feature is, in our understanding, a prominent or distinctive user-recognisable aspects, quality, or characteristics of a system that is related to a specific set of requirements, whose realisation enable the feature [12, 31]. In addition to features, we specify an <i>Use Case Overview</i> , i.e. a (potentially) graphical overview of the use cases specified in full in the usage model.
Usage Model	Structured text, UML activity diagrams	This content item details the <i>Use Case Overview</i> of the <i>System Vision</i> in its <i>Use Cases</i> . We distinguish <i>Services</i> and <i>UML activity Use Cases</i> . Both concepts are means to describe (black box) system behaviour. <i>Use Cases</i> describe sequences of interaction between <i>Actors</i> (<i>realising</i> user groups) and the system as a whole. More precisely, a use case represents a collection of interaction scenarios, each defining a set of interrelated actions that either are executed by an actor or by the system under consideration [14]. For each use case, there is at least one <i>Functional Scenario</i> in which <i>Actors</i> participate. A Scenario inherits from a requirement (not a whole use case) and each Scenario is detailed into Actions, which can be <i>Actor Actions</i> or <i>System Actions</i> each processing <i>Data Objects</i> . Functional scenarios are triggered by Events. Furthermore, we include <i>Generic Scenarios</i> , which serve for the satisfaction of <i>Quality Requirements</i> as they provide a means to specify generic interactions between actors and a system not necessarily motivated by business processes, such as maintenance activities an administrator performs.
Service Model	Graphs	This content item is relevant for the domain of business information systems and specifies <i>Services</i> with a <i>Collaboration Contract</i> and a defined <i>Quality of Service</i> . <i>Services</i> describe a logical representation of a use case, not necessarily involving actors or concrete sequences of interaction. A service represents user-visible functions that the system shall offer and is described via input/output-relations [25, 59], i.e., the mapping of (typed) inputs and outputs, both represented as information system objects. The quality of service is described using <i>Service Parameters</i> (corresponding to <i>Metrics</i> from the <i>Quality Requirements</i>) that have particular <i>Service Levels</i> . If a collaboration contract specifies a set of service (calls) as part of a causal relation, we use scenarios within the <i>Functional Scenarios</i> of the <i>Usage Model</i> .
Data Model	mboxUML class diagrams	The <i>Data Model</i> contains all objects processed as part of functions and interaction scenarios.

Functional Hierarchy	Graphs, I/O-tables	The <i>User-visible Functions</i> equal the <i>Services</i> from the BIS-specific <i>Service Model</i> and realise the <i>System Actions</i> from the <i>Usage Model</i> . Functions are organised in a hierarchy to build the transition to the system specification as they describe a logical representation of a system action and offered by typed interfaces. When a Function is triggered, we can optionally define <i>Modes</i> , which we use in the <i>System Specification</i> for the definition of detailed <i>Behaviour Models</i> . <i>Services</i> , in turn, describe a complementary logical representation of a use case, not necessarily involving actors or concrete sequences of interaction. They furthermore offer <i>Interfaces</i> , which are typed according to the <i>Data Objects</i> of the <i>Data Model</i> .
Quality Requirements	Natural text	Quality requirements are assessed by <i>Measurements</i> that can be either a <i>Normative Reference</i> (e.g. a GUI style guide) or a <i>Metric</i> . Quality Requirements constrain <i>System Actions</i> and can be satisfied by <i>Generic Scenarios</i> . We make use of quality definition models as by Deissenböck et al. [16].
Deployment Requirements	Natural text	<i>Deployment Requirements</i> describe demands towards the deployment procedure, constraining the process design of the deployment, and the technical environment during initial launch of the system or specific parts of it.
System Constraints	Natural text	The system's constraints describe logical and technical restrictions on a system's architecture, its functionality by means of single atomic actions, and its quality by means of assessable system quality requirements. We consider concepts that describe the transition to logical and technical architecture layers acc. to [69]. Hence, we see a system as a grey box rather than as a glass box, since we restrict systems' internals, but do not consider their logical structure by interacting components, interface specifications, and functions.
Process Requirements	Natural text	<i>Process Requirements</i> constrain the content and / or structure of selected artefact types and the process model, i.e., the definition of the milestones regarding time schedules, used infrastructure like mandatory tools, and compliance to selected standards and software process models.
Risk List	Natural text	The <i>Risk List</i> includes a description of all risks that are related to project-specific requirements. The conceptualisation of requirements risks is considered on the basis of an artefact model [28, 29]. The <i>Requirements Risks</i> are implied by the various types of Requirements and we use the risk list as an interface to risk management. Each risk is caused by a <i>Risk Factor</i> . A <i>Risk Trend</i> is composed by all Risk Factors.

A.3 System Specification

The system specification is depicted in Fig. 16. It contains the Function Model, the Component Model, the Behaviour Model, the State Model, and the Data Model. A description of the content items is provided in Table 8.

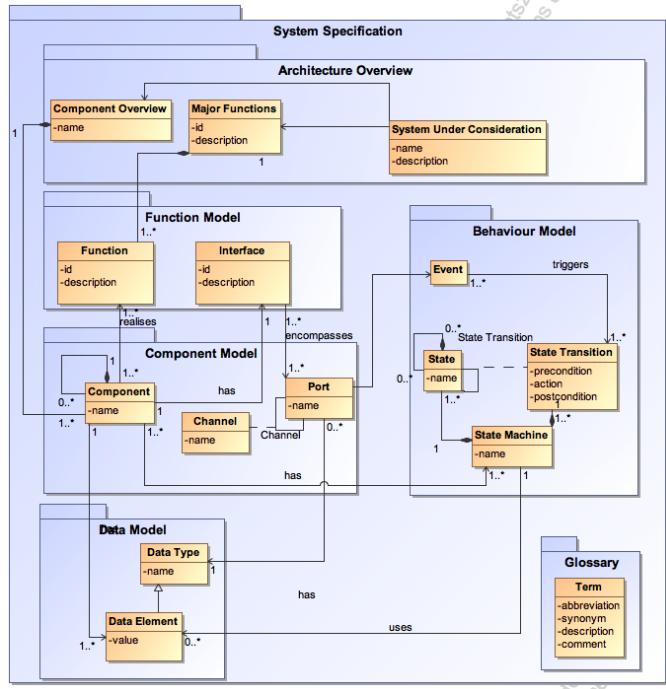


Fig. 16: The AMDiRE system specification.

Table 8: Content items in the System Specification.

Content Item	Exempl. Notation	Description
Architecture Overview	Component diagram	The <i>Architecture Overview</i> includes the <i>Component Overview</i> of the <i>Components</i> as well as the <i>Major Functions</i> that summarize the <i>Functional Hierarchy</i> .
Function Model	Graphs, tables	The <i>System Functions</i> offered by the <i>System Interface</i> realise the <i>User-visible Functions</i> from the <i>Functional Hierarchy</i> . The system interface encompasses the Ports detailed in the Component Model.

Component Model	Component diagram	The component model describes the logical component architecture in detail including <i>Ports</i> (building typed interfaces) and <i>Channels</i> over which components communicate. Components can be decomposed into more further sub-components [10].
Behaviour Model	Automata	The behaviour of components is specified with by <i>Events</i> , <i>States</i> and <i>State Transitions</i> . The resulting state machines specify the behaviour and the used data elements of the data model.
Data Model	UML class diagram, dictionary	We specify data elements of a certain type and their relations as they result from the behaviour model. To this end, <i>Data Elements</i> refine <i>Data Objects</i> from the requirements specification and have a particular <i>Data Type</i> .

References

1. Avison D, Fitzgerald G (2006) Information Systems Development: Methodologies, Techniques, and Tools, 4th edn. McGraw-Hill Education, Berkshire, United Kingdom
2. Berenbach B, Paulish D, Kazmeier J, Rudorfer A (2009) Software & Systems Requirements Engineering: In Practice. ISBN-13: 978-0071605472, McGraw-Hill Osborne Media
3. Besner V (2013) Bsc. thesis: Evaluation of an artefact-based requirements engineering approach. Master's thesis, Technische Universität München, Department of Informatics
4. Boege J (2008) A New Standard for Quality Requirements. IEEE Software 25(2):57–63
5. Boehm B, Brown J, Kaspar H, Lipow M, Macleod G, Merrit M (1978) Characteristics of Software Quality. ISBN-13: 978-0444851055, Elsevier Science Ltd North-Holland Pub. Co.
6. Braun C, Wortmann F, Hafner M, Winter R (2005) Method Construction - A Core Approach to Organizational Engineering. In: Proceedings of the 20th ACM symposium on Applied computing (SAC '05), ACM New York, NY, USA, pp 1295–1299
7. Braun P, Broy M, Houdek F, Kirchmayr M, Müller M, Penzenstadler B, Pohl K, Weyer T (2010) Guiding requirements engineering for software-intensive embedded systems in the automotive industry. Computer Science Research and Development
8. Brinkkemper S (1996) Method Engineering: Engineering of Information Systems Development Methods and Tools. Information and Software Technology 38(4):275–280
9. Broy M (2006) Requirements Engineering as a Key to Holistic Software Quality. In: Levi A, Savas E, Yenigun H, Balcisoy S, Saygin Y (eds) Proceedings of the 21th International Symposium on Computer and Information Sciences (ISCIS 2006), Springer-Verlag Berlin, vol 4263, pp 24–34
10. Broy M, Feilkas M, Grünbauer J, Gruler A, Harhurin A, Hartmann J, Penzenstadler B, Schätz B, Wild D (2008) Umfassendes Architekturmodell für das Engineering eingebetteter Software-intensiver Systeme. Tech. Rep. TUM-I0816, Technische Universität München
11. Cavano J, McCall J (1978) A Framework for the Measurement of Software Quality. SIGSOFT Software Engineering Notes 3(5):133–139, DOI <http://doi.acm.org/10.1145/800283.811113>
12. Classen A, Heymans P, Schobbens PY (2008) What's in a Feature : A Requirements Engineering Perspective. In: Fiadeiro J, Inverardi P (eds) Proceeding of the 11th International Conference on Fundamental Approaches to Software Engineering (FASE 08) in Conjunction with ETAPS 08, Springer-Verlag Berlin, no. 4961 in FASE/ETAPS, pp 16–30
13. Cleland D (ed) (2005) Project Management Field Guide. ISBN-13: 978-0471292067, John Wiley and Sons, Inc.

14. Cockburn A (2000) Writing Effective Use Cases. ISBN-13: 978-0201702255, Addison-Wesley Longman Publishing Co., Inc.
15. Deissenboeck F (2009) Continuous Quality Control of Long-Lived Software Systems. PhD thesis, Technische Universität München, Department of Informatics
16. Deissenboeck F, Juergens E, Lochmann K, Wagner S (2009) Software Quality Models: Purposes, Usage Scenarios and Requirements. In: Proceedings of the 7th international workshop on Software Quality (WoSQ 09), IEEE Computer Society Press, p N/A
17. Doerr J, Kerkow D, Von Knethen A, Paech B (2003) Eliciting Efficiency Requirements with Use Cases. In: Proceedings of the 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ 03), pp 23–32
18. Eveleens J, Verhoef T (2010) The Rise and Fall of the Chaos Report Figures. *IEEE Software* 27(1):30–36
19. Foorthuis R, Brinkkemper S, Bos R (2009) An Artifact Model for Projects Conforming to Enterprise Architecture. In: Stirna J, Persson A (eds) Proceedings of the 1st Working Conference on the Practice of Enterprise Modeling (POEM), Springer-Verlag, vol 15, pp 30–46
20. Garvin D (1984) What does Product Quality really mean? *MIT Sloan Management Review* 26(1):25–43
21. Geisberger E, Broy M, Berenbach B, Kazmeier J, Paulish D, Rudorfer A (2006) Requirements Engineering Reference Model (REM). Tech. Rep. TUM-I0618, Technische Universität München
22. Glinz M (2007) On Non-Functional Requirements. In: Proceedings of the 15th IEEE International Conference on Requirements Engineering (RE 07), IEEE Computer Society, pp 21–26
23. Gonzalez-Perez C, Henderson-Sellers B (2006) A Powertype-based Metamodelling Framework. *Software and Systems Modeling* 5(1):72–90
24. Hevner AR, March ST, Park J, Ram S (2004) Design science in information systems research. *MIS Quarterly* 28(1):pp. 75–105, URL <http://www.jstor.org/stable/25148625>
25. Hummel B, Thyssen J (2009) Behavioural Specification of Reactive Systems Using Stream-Based I/O Tables. In: Proceedings of the 7th IEEE International Conference on Software Engineering and Formal Methods (SEFM 09), IEEE Computer Society, pp 137–146
26. IEEE (1998) IEEE Recommended Practice for Software Requirements Specifications – IEEE Std 830-1998. Technical Standard IEEE Std 830-1998, The Institute of Electrical and Electronics Engineers, Inc.
27. IIBA (2009) A Guide to the Business Analysis Body of Knowledge (BABOK Guide). ISBN-13: 978-0981129211, International Institute of Business Analysis (IIBA)
28. Islam S (2009) Software Development Risk Management Model - A Goal Driven Approach. In: Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundation of Software Engineering (ESEC / FSE), ACM

- Press, New York, NY, USA, pp 5–8
- 29. Islam S, Houmb S, Mendez Fernandez D, Joarder M (2009) Offshore-Outsourced Software Development Risk Management Model. In: Proceedings of the 12th IEEE International Conference on Computer and Information Technology (ICCIT 09), pp 514–519
 - 30. ISO/IEC (2007) ISO / IEC Std. 24744 Software Engineering - Metamodel for Development Methodologies. Tech. Rep. 2007-02-15, International Organization for Standardization
 - 31. Kang K, Cohen S, Hess J, Nowak W, Peterson S (1990) Feature-Oriented Domain Analysis (FODA) Feasibility Study. Tech. Rep. CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University Pittsburgh, PA.
 - 32. Kitchenham B, Pfleeger S (1996) Software Quality: The Elusive Target. *IEEE Software* 13(1):12–21
 - 33. Kroll P, Kruchten P (2003) The Rational Unified Process made Easy: a Practitioner's Guide to the RUP. ISBN-13: 978-0321166098, Addison-Wesley Longman Publishing Co., Inc.
 - 34. Kuhrmann M, Mendez Fernandez D, Steenweg R (2013) Systematic Software Process Development - Where do we stand today? In: Proceedings of the 9th International Conference on Software and System Process (ICSSP 13, ACM Press
 - 35. van Lamsweerde A (2009) Requirements Engineering: From System Goals to UML Models to Software Specifications. ISBN-13: 978-0470012703, Wiley & Sons
 - 36. List B, Korherr B (2006) An Evaluation of Conceptual Business Process Modelling Languages. In: Proceedings of the 2006 ACM symposium on Applied computing (SAC 06), ACM New York, NY, USA, pp 1532–1539
 - 37. Mendez D, Penzenstadler B, Eckhardt J (2013) Amdire online resources. <http://www4.in.tum.de/~mendezfe/openspace.shtml>
 - 38. Mendez Fernandez D (2011) Requirements Engineering: Artefact-Based Customisation. PhD thesis, Technische Universität München, Department of Informatics
 - 39. Mendez Fernandez D, Kuhrmann M (2009) Artefact-Based Requirements Engineering and its Integration into a Process Framework: A Customisable Model-based Approach for Business Information Systems' Analysis. Tech. Rep. TUM-I0929, Technische Universität München, URL <http://www.in.tum.de/forschung/pub/reports/2006/TUM-I0618.pdf.gz>
 - 40. Mendez Fernandez D, Wagner S (2013) Naming the Pain in Requirements Engineering: Design of a global Family of Surveys and first Results from Germany. In: Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering (EASE 2013, ACM Press, pp 183–194
 - 41. Mendez Fernandez D, Penzenstadler B, Kuhrmann M, Broy M (2010) A Meta Model for Artefact-Orientation: Fundamentals and Lessons Learned in Requirements Engineering. In: Petriu D, Rouquette N, Haugen O (eds) Proceedings of the 13th International Conference on Model Driven Engi-

- neering Languages and Systems (Models), Springer-Verlag Berlin Heidelberg, vol 6395, pp 183–197
42. Mendez Fernandez D, Lochmann K, Penzenstadler B, Wagner S (2011) A Case Study on the Application of an Artefact-Based Requirements Engineering Approach. In: Proceedings of the 15th International Conference on Evaluation and Assessment in Software Engineering (EASE 2011), Institution of Engineering and Technology (IET), pp 104–113
 43. Mendez Fernandez D, Wagner S, Lochmann K, Baumann A, de Carne H (2012) Field Study on Requirements Engineering: Investigation of Artefacts, Project Parameters, and Execution Strategies. *Information and Software Technology* 54(2):162–178
 44. Mendez Fernandez, D and Wagner, S (2013) Naming the Pain in Requirements Engineering. *Information and Software Technology Currently in Review*, Reviewer Version available under: <http://www4.in.tum.de/~mendezfe/openspace/NaPiRE/INFSOF-S-13-00428.pdf>
 45. Mendez Fernandez, D and Wieringa, R (2013) Improving requirements engineering by artefact orientation. In: Proc. of the 14th International Conference on Product-Focused Software Development and Process Improvement (PROFES '13), Springer, vol 108–122
 46. Merriam Webster Online Dictionary (2014) Definition: Process. <http://www.merriam-webster.com/dictionary/process>
 47. Milne A, Maiden N (2011) Power and Politics in Requirements Engineering: A Proposed Research Agenda. In: Proc. 19th International Conference on Requirements Engineering (RE 2011), IEEE Computer Society
 48. Nuseibeh B, Easterbrook S (2000) Requirements Engineering: A Roadmap. In: Proceedings of the Conference on the Future of Software Engineering, ACM, New York, NY, USA, pp 35–46
 49. OMG (2008) Software and Systems Process Engineering Meta-Model (SPEM) Specification V. 2.0. Technical Standard formal/2008-04-01, Object Management Group
 50. Parnas DL, Clements PC (1986) A Rational Design Process: How and Why to fake it. *IEEE Transactions on Software Engineering* 12(2):251–257
 51. Pedreira O, Piattini M, Luaces M, Brisaboa N (2007) A Systematic Review of Software Process Tailoring. *SIGSOFT Software Engineering Notes* 32(3):1–6
 52. Penzenstadler B, Eckhardt J (2012) A requirements engineering content model for cyber-physical systems. In: 20th International Conference on Requirements Engineering
 53. Penzenstadler B, Sikora E, Pohl K (2009) A Requirements Reference Model for Model-based Requirements Engineering in the Automotive Domain. In: Glinz M, Heymans P (eds) Proceedings of the The 15th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 09), Springer-Verlag Berlin Heidelberg, vol 5512, pp 212–217
 54. Penzenstadler B, Eckhardt J, Mendez Fernandez D (2013) Two Replication Studies for Evaluating Artefact Models in RE: Results and Lessons

- Learnt. In: To appear in: Proc. of the 3rd International Workshop on Replication in Empirical Software Engineering Research (RESER 13)
- 55. Penzenstadler B, Mendez Fernandez D, Eckhardt J (2013) Understanding the Impact of Artefact-based RE - Design of a Replication Study. In: To appear in: Proc. of the 7th International Symposium on Empirical Software Engineering and Measurement (ESEM 13)
 - 56. Regnell B, Paech B, Aurum A, Wohlin C, Dutoit A, och Dag J (2001) Requirements Mean Decisions!–Research Issues for Understanding and Supporting Decision-Making in Requirements Engineering. In: Bengtsson P (ed) Proceedings of the 1st Swedish Conference on Software Engineering Research and Practise, Bleking Institute of Technology, no. 2001:10 (Research Report) in SERP, pp 49–52
 - 57. Regnell B, Svensson RB, Wnuk K (2008) Can We Beat the Complexity of Very Large-Scale Requirements Engineering? In: Proceedings of the 14th international conference on Requirements Engineering: Foundation for Software Quality, Springer-Verlag, REFSQ '08, pp 123–128
 - 58. Ritter F, Schooler LJ (2001) The learning curve, Oxford, UK: Pergamon
 - 59. Rittmann S (2008) A Methodology for Modeling Usage Behavior of Multi-functional Systems. PhD thesis, Technische Universität München, Department of Informatics
 - 60. Robertson J, Robertson S (2007) Volere Requirements Specification Templates - Edition 11. <http://www.volere.co.uk>
 - 61. Rogers EM (1995) Diffusion of Innovation, 4th edn. The Free Press
 - 62. Schätz B (2008) Model-Based Development of Software Systems: From Models to Tools. Habilitationsschrift, Technische Universität München, Department of Informatics
 - 63. Schätz B, Fleischmann A, Geisberger E, Pister M (2005) Model-Based Requirements Engineering with AutoRAID. In: Cremers A, Manthey R, Martini P, Steinhage V (eds) Proceedings zum Workshop Modellbasierte Qualitätssicherung (QUAM), Bonner Köllen Verlag, Lecture Notes in Informatics (LNI), pp 511–516
 - 64. Silva M, Oliveira T (2011) Towards Detailed Software Artifact Specification with SPEMArti. In: Proceedings of the 7th International Conference on Software and System Process (ICSSP 13)
 - 65. Spence AM (1981) The learning curve and competition. *The Bell Journal of Economics* 12(1):pp. 49–70
 - 66. Standish Group International (1995) The Chaos Report. <http://net.educause.edu/ir/library/pdf/NCP08083B.pdf>, accessed on 2009/03/15
 - 67. Tell P, Babar M (2012) Activity Theory applied to Global Software Engineering: Theoretical Foundations and Implications for Tool Builders. In: Proceedings of the 7th International Conference on Global Software Engineering, pp 21–30
 - 68. Ter Hofstede A, Verhoef T (1997) On the Feasibility of Situational Method Engineering* 1. *Information Systems* 22(6/7):401–422, URL <http://linkinghub.elsevier.com/retrieve/pii/S0306437997000240>

69. Wiegers K (2003) Software Requirements, 2nd edn. ISBN-13: 978-0735618794, Microsoft Press, Redmond, WA, USA
70. Wieringa R (2009) Design Science as Nested Problem Solving. In: Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology, ACM New York, NY, USA
71. Wieringa R, Aycse M (2012) Technical action research as a validation method in information systems design science. In: Proceedings of the 7th international conference on Design Science Research in Information Systems: advances in theory and practice, pp 220–238
72. Wieringa RJ (2010) Relevance and problem choice in design science. In: Global Perspectives on Design Science Research., Springer, pp 61–76
73. Wiesi S (2013) Msc. thesis: Development and evaluation of an artefact-based requirements engineering approach for agile development. Master's thesis, Technische Universität München, Department of Informatics
74. Zave P (1997) Classification of research efforts in requirements engineering. ACM Computing Surveys Vol. 29(4):315–321