Abstract

Traditional software engineering and information systems engineering is structured into requirements analysis and definition, systems design, systems implementation and testing, and systems operation and maintenance. For web information systems the traditional approach suffers in three obstacles: late integration of architectural decisions, neglecting user expectations, and late implementations. Web information systems follow pre-defined three-tiered architectures. They also must consider expectations, profiles and portfolio of a large variety of users. Additionally, users expect an early involvement into the development and an early evaluation of the system. At the same time, traditional software engineering has led to quality assurance during the development process based on CMMI and SPICE. This paper shows how the achievements of traditional software engineering can be preserved while overcoming the obstacles. We develop an approach that integrates application domain description with development of presentation and information systems. The development methodology is improved based on the requirements of SPICE.

1. Introduction

1.1 Web Information Systems

In general, every data-intensive information system that is realised in a way that users can access it via web browsers or thin clients will be called a web information system (WIS). Typically, such systems are realised by a web site, i.e. a collection of web pages, each of which is just a file that can be interpreted by the browser. We may thus apply classical methods of information systems engineering to website development. The main difference is however that the purpose and usage of the systems as well as the users are known in advance. Therefore, we have to pay more attention to a “mission statement” for the system, i.e. the important questions “Who will use the system?”, “Which user intentions and behaviour shall be supported?”, “Which technical devices will be used by the users?”, etc. have to be taken into account. We thus consider six different aspects for web information systems: intention, usage, content, functionality, context, and presentation.

1.2 Engineering of Web Information Systems

Björner [2] and Heinrich [6] consider software engineering on the basis of the trilogy consisting of the application domain description, the requirements prescriptions, and finally the systems specifications. They extend modern software engineering approaches, e.g. [11, 15] by explicit consideration of the application domain. Schewe and Thalheim [16, 19] extend these approaches by (1) explicit consideration of user expectations, profiles and portfolio and by (2) storyboards and story spaces. WIS specification is oriented towards systems that are easy and intuitive to use. Therefore the classical information systems development model adds to systems development the development of presentation systems, too. Classically user interfaces are built after the system has been developed. In this case, the user has to learn how the system behaves and must adapt his/her behaviour to the systems behaviour. We repair this mismatch by primarily considering the user worlds, the user stories, and the applications.

WIS has two different faces: the systems perspective and the user perspective. These perspectives are tightly related to each other. We consider the presentation system as an integral part of WIS. It satisfies all user requirements, and it is based on real life cases.

Top-down development of systems seems to be the most appropriate whenever a system is developed from scratch or a system is extended. For this reason, we may differentiate among three layers: the systems description
and prescription layer, the conceptual specification layer, and
the systems layer. These layers may be extended by
the strategic layer that describes the general intention of
the system, by the business user layer that describes how
business users will see the system and by the logical layer
that relates the conceptual layer to the systems layer by
using the systems languages for programming and
specification.

Classical software engineering typically climbs down
to the implementation layer in order to create a productive
system. The usual way in today’s WIS development is a
manual approach: human modelling experts interpret the
specification to enrich and transform it. This way of
developing specifications is error-prone: even if the
specification on a certain layer is given in a formal
language, the modelling expert as a human being will not
interpret it in a formal way. Misinterpretations,
misunderstandings, and therefore the loss of already
specified system properties is the usual business.

To overcome these obstacles a methodology for WIS
engineering (WIS-E) has been developed [19]. The
methodology is under continuous development. To
prioritize the efforts and to ensure coherence of the
enhancements, a comparison with more mature software
engineering approaches was considered to be of benefit.

1.3 Software process improvement

Approaches to improve a software process can be
based on e.g. modelling, assessment, measurement,
technology adoption or management. The approaches
supplement each other, but one of them is usually in a
dominating position. Process assessment is a norm-based
approach, which usually leads to evolutionary process
improvement [1].

The development of web information systems is a
special case for a software process. Such a process is
considered to be the set of activities, methods, and
practices used in the production and evolution of software
[7] and the associated products [12]. A process transforms
its inputs to outputs. The properties of the process have a
great influence on its efficiency and on the properties of
its outputs. It is widely accepted that the quality of a
software product is largely determined by the quality of
the process used [21]. The quality of a process can be
measured using the attributes of process capability.

Since 1998 the technical report ISO/IEC TR 15504 [9]
has provided a framework for the assessment of software
processes. It is generally known as SPICE (Software
Process Improvement and Capability dEtermination), a
term, which originally stands for the initiative to support
the development of a standard for software process
assessment. The international standard ISO/IEC 15504,
with five parts, has been published in 2003-2006 [9, 10].

Process capability is a characterization of the ability of
a process to meet current or projected business goals [8].
Part 2 of the standard defines a measurement framework
for the assessment of process capability. In the
framework, Process capability is defined on a six point
ordinal scale (0-5) that enables capability to be assessed
from Incomplete (0) to Optimizing (5). The scale
represents increasing capability of the implemented
process. The measure of capability is based upon a set of
process attributes, which define particular aspects of
process capability [9].

1.4 Scope of the study

This paper introduces the background for WIS
engineering and its improvement using SPICE. WIS must
provide a sophisticated support for a large variety of
users, a large variety of usage stories, and for different
(technical) environments. Due to this flexibility the
development of WIS differs from the development of
information systems by careful elaboration of the
application domain, by adaptation to users, stories,
environments etc. In this context, process assessments
and interpretation of the software engineering process
models are used to improve a WIS engineering
methodology.

The next section explains the process of methodology
improvement and the general approach. Section 3
describes one of the major novel steps for WIS
engineering based on the SPICE analysis: The application
domain description. Section 4 concludes our experiences
in supporting WIS engineering with SPICE assessments.

2. Towards managed Web Information
Systems engineering

2.1 Process capability in process improvement
context

The starting point for software process improvement
actions is the gap between the current state of an
organization and the desired future state [5]. These two
states can be characterized using a norm for good
software practices. The most popular among such norms
are the Software Engineering Institute’s CMMI [3] and
SPICE [10] by international standardization bodies
ISO/IEC. The current state of software practice can be
evaluated with the help of process assessments, which is a
disciplined appraisal of an organization’s software
process by utilizing a norm: the software process
assessment model.

2.2 Applying process improvement principles in
method engineering
A methodology for WIS engineering (WIS-E) was evaluated using a standard assessment approach with the aim to generate improvement ideas for the development of the methodology. The purpose of using an assessment approach was to improve the structure of WIS-E and its practice definitions. The assessment was performed (1) to check that corresponding elements of 15504-5 process assessment model can be found in WIS-E; (2) to ensure that future use of WIS-E would satisfy the requirements of the assessment model and; (3) to check the assessment model scope from a database development oriented point of view.

The assessment was performed in several phases during a long period of time. The assessment team consisted of the Co-Design chief engineer and two experienced assessors. The main steps of the assessment were planning, examination, discussion, documentation and review. During the planning, the scope of the assessment was fixed and the necessary material selected. The material that described the WIS-E was examined by the assessors and detailed questions were prepared. In the discussions within the assessment team the methodology was studied and the assessment questions were answered. The documentation of the assessment results provided a set of improvement ideas and a rough indication of the methodology’s correspondence to the capability requirements of the assessment model. During the review the assessment results were explained and the findings were discussed to create a consistent path for improvement and to set priorities.

2.3 Experiences of using SPICE in developing a methodology

In general, the assessment approach worked well and the findings were considered relevant. A difficulty in the assessment was that the methodology was developed at the same time. The extended assessment period and the pressure to modify the methodology based on the preliminary discussions, made some findings obsolete.

In the beginning of the assessment the main issue was to map the Co-Design processes to the processes of the 15504-5 assessment model. E.g. the Stakeholder contract step contained elements of three model processes i.e. Requirements elicitation, Contract agreement and Project management. Besides the structural issues, also the content of the WIS-E practices and work products was examined and discussed.

During the reviews concepts were clarified and redundant descriptive elements removed, also the wording of the elements was harmonized. The main benefit for WIS-E was the improvement of the structure. The result can be seen in clarified statements of process purposes and outcomes, and defined work products. A structured methodology is easy to develop further, the completeness and correctness of the methodology can be ensured, and the overall quality of the method is improved.

When considering the content of the methodology the software engineering process assessment approach provided only minor improvements. On the contrary, the assessment model could be further developed based on the Co-Design approach. The most prominent areas are the database design and the consideration of the user needs in wider scope and from various aspects, than what the prevailing software engineering process models cover.

2.4 Implications for WIS engineering

We applied the SPICE framework for improving WIS engineering, for systematic development of all facets, aspects, and work products, and for orchestration of WIS specifications. Web information systems engineering models the activities performed or managed by people participating in the engineering process. Models support understanding of the application domain and of the WIS, can be a source of inspirations, can be used for presentation and training, are abstractions that assert and predict behaviour of the WIS, and are the source for the implementation of the web information system.

Fig. 1. The primary dimensions of information systems engineering

Activities may result in work products, may revise existing work products, or may be based on different work products. Work products considered in our framework are the developed documents and their role to the completion of the activity (postcondition).

We therefore distinguish between the five primary dimensions of WIS engineering. Activities (“how”) describe the way how the work is performed and the practises for the work. Work products (“what”) are the
result of the specification and are used during specification. **Roles** (“who”) describe obligations and permissions, or the involvement of actors in the specification process. **Aspects** (“where”) are used for separation of concern during the specification process. **Resources** (“on which basis”) are the basis for the specification. These five main dimensions are hierarchically structured. We structure the associations among the dimensions in Figure 1.

These five dimensions must be mapped to technology. This mapping includes the derivation of the general solution and of the software architecture. The five dimensions can be enhanced by secondary dimensions. Finally, each object of one dimension may be associated to other objects of the same or other dimensions. We observe in the resource dimension, that, for instance, methods are founded on theories.

The Co-design approach to modelling [16, 19] integrates specification of **structuring** (structure and static integrity constraints), **functionality** (processes and dynamic integrity constraints), **distribution** (services and exchange frames), and **interactivity** (foreseen stories and envisioned actors).

### 2.5 Orchestration of WIS development for managed engineering

Orchestration uses the metaphor to music. It denotes the arrangement of a musical composition for performance by an orchestra. It may also denote harmonious organization, i.e. through orchestration of cultural diversities. Figure 1 displays dimensions of information systems engineering. Partners must integrate all activities, for all aspects, for all resources, and all work products. **SPICE** [9] requires for development processes that the implemented process achieves its process purpose (SPICE level 1). **SPICE** level 2 is achieved if the process is implemented in a managed fashion (planned, monitored and adjusted) and its work products are appropriately established, controlled and maintained.

Therefore, managed engineering is based on performance management and on work product management. Performance management requires that objectives for the performance of the process are identified, performance of the process is planned and monitored, performance of the process is adjusted to meet plans, responsibilities and authorities for performing the process are defined, assigned and communicated, resources and information necessary for performing the process are identified, made available, allocated and used, and interfaces between the involved parties are managed to ensure both effective communication and also clear assignment of responsibility. Work product management requires a well-defined and well-implemented work product management. Requirements for the work products of the process must be defined as well as requirements for documentation and control of the work. Work products must be appropriately identified, documented, and controlled. Work products are going to be reviewed in accordance with planned arrangements and adjusted as necessary to meet requirements. [10]

We show now in the sequel how orchestration of WIS development leads to managed WIS engineering. Due to space limitations we restrict on the first process: Application domain description and requirements statement.

### 3. Application domain description for WIS

#### 3.1 The Co-Design framework for WIS engineering

The co-design framework [18, 20] provides a methodology for description, prescription and specification. A methodology is the study of and knowledge about methods. Methods used in the co-design framework are based

- (a) on the extended entity-relationship model that supports specification of structures and functions,
- (b) on the storyboarding language SiteLang that supports specification of users, their profiles, their portfolio, their actions, and their stories and
- (c) on the framework DistLang that provides a general framework for architecturing, for development of services and exchange frames.

We distinguish a number of facets or views on the application domain. Typical facets to be considered are business procedure and rule facets, intrinsic facets, support technology facets, management and organisation facets, script facets, and human behaviour.

In the sequel we demonstrate how SPICE-improved WIS engineering can be performed for the WIS application domain engineering. Application domain engineering forms a process in the sense of **SPICE**. It aims in describing the application perspective, i.e. the subject world, the usage world, the intentions of the WIS according to **Pohl** and **Rolland** [13, 14]. It results in a general statement of requirements. Requirements engineering aims in elicitation of requirements within the system environment, exploration of system choices, complete extraction of functional and non-functional requirements, conflict detection and resolution, documentation and negotiation of agreed requirements, and in providing a framework for WIS evolution.

Classical application domain engineering is rather informal and fuzzy. It is based on general multi-faceted descriptions of desires, beliefs, intentions, skills, and
awareness [14]. The co-design framework is the first one that results in a formal description of the application domain. At the same time, the framework is based on a methodology. We describe the first process, Application domain description and requirements statement, of this methodology in the next subsection.

3.2 Application domain description and requirements statement

The most important outcome of application domain engineering is an application domain model and its associated application domain theory. The main activity is to gather from application domain business users, from literature and from our observations the knowledge about the application domain. It is combined with validation, i.e. the assurance that the application domain description commensurates with how the business users view the application domain. It also includes the application domain analysis, i.e. the study of application domain (rough) statements, the discovery of potential inconsistencies, conflicts and incompleteness with the aim of forming concepts from these statements.

**Process purpose: Goals and subject**

Each process of WIS development must be characterised by goals and the subject.

<table>
<thead>
<tr>
<th>Goals and subject</th>
<th>Application domain description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Agreement for development</td>
</tr>
<tr>
<td></td>
<td>Project scope: Milestones, financial issues</td>
</tr>
<tr>
<td></td>
<td>Clarification of development goals (intentions, rationale)</td>
</tr>
<tr>
<td></td>
<td>Sketch of requirements</td>
</tr>
</tbody>
</table>

**Process Outcomes: Work Products as Process Results**

The work product is a result or deliverable of the execution of a process and includes services, systems (software and hardware) and processed materials. It has elements that satisfy one or more aspects of a process purpose and may be represented on various media (tangible and intangible).

Documents of the application domain layer are HERM [18] concept maps, HERM functionality feature descriptions, the DistrLang distribution specification resulting in contract sketches which include quality criteria, and the SiteLang interactivity specification of the application story with main application steps. The documents are combined within the Stakeholder contract and the feasibility study. Additionally a number of internal documents are developed such as life case studies, description of intentions, context specification, and user profiles and portfolio.

**Table 1. Process purpose**

<table>
<thead>
<tr>
<th>Goals and subject</th>
<th>Application domain description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Agreement for development</td>
</tr>
<tr>
<td></td>
<td>Project scope: Milestones, financial issues</td>
</tr>
<tr>
<td></td>
<td>Clarification of development goals (intentions, rationale)</td>
</tr>
<tr>
<td></td>
<td>Sketch of requirements</td>
</tr>
</tbody>
</table>

**Internal section**

<table>
<thead>
<tr>
<th>Goals and subject</th>
<th>Application domain description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Agreement for development</td>
</tr>
<tr>
<td></td>
<td>Project scope: Milestones, financial issues</td>
</tr>
<tr>
<td></td>
<td>Clarification of development goals (intentions, rationale)</td>
</tr>
<tr>
<td></td>
<td>Sketch of requirements</td>
</tr>
</tbody>
</table>

**Practises: Base Activities and Steps**

The system development process is usually decomposed into clearly distinguishable development activities that support to tackle separate issues at separate time. An activity either starts from scratch and results in a completed work product or starts from a set of work products and results in a completed or revised work product. Activities are work product transformations. Therefore, the steps can be given through refinement activities, i.e., elicitation, determination, instantiation, extension, and fitting. Activities typically proceed in one or (usually) more steps. Activities are describing the ways how the work is performed and can be practices...
according to SPICE. A practice contributes to achieving a specific development process purpose or to the achievement of a specific development process attribute or enhances the capability of a development process. Activities consist of steps. These activities are enhanced by acquisition activities and activities supporting the life cycle, which are not described here.

We envision three base activities and structure them into development steps:
1. Development of application domain description
2. Development of the stakeholder contract
3. Development of the internal documents such as description of product components and quality requirements with base activities, generic practices and capabilities and estimation of efforts etc.

We demonstrate the base activities for the first one in Table 3. We use the SiteLang specification language [17] and the conceptual framework of [16].

<table>
<thead>
<tr>
<th>Development of application domain description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Analyze strategic information</td>
</tr>
<tr>
<td>Specify mission and brand</td>
</tr>
<tr>
<td>Characterise in general tasks and users of</td>
</tr>
<tr>
<td>application</td>
</tr>
<tr>
<td>Characterise in general content and functions</td>
</tr>
<tr>
<td>domain</td>
</tr>
<tr>
<td>Describe WIS context</td>
</tr>
<tr>
<td>2. Derive intentions of the WIS,</td>
</tr>
<tr>
<td>Obtain general requirements</td>
</tr>
<tr>
<td>Extract life cases</td>
</tr>
<tr>
<td>Describe scenarios, scenes, actions, and context</td>
</tr>
<tr>
<td>Describe user profiles and user portfolio</td>
</tr>
<tr>
<td>Derive actor profiles and actor portfolio, personas</td>
</tr>
<tr>
<td>3. Extract business rule model and storyboards</td>
</tr>
<tr>
<td>Develop scenarios, scenes, and actions</td>
</tr>
<tr>
<td>Specify life cases and context</td>
</tr>
<tr>
<td>Eliciting metaphors</td>
</tr>
<tr>
<td>4. Revise business rules of the application</td>
</tr>
<tr>
<td>Possibly with reorganization models</td>
</tr>
<tr>
<td>5. Compare new business rules with known visions</td>
</tr>
<tr>
<td>6. Compare with products of competitors</td>
</tr>
<tr>
<td>7. Derive WIS utilisation</td>
</tr>
<tr>
<td>Describe scenarios to be supported</td>
</tr>
<tr>
<td>Describe activities based on word fields</td>
</tr>
<tr>
<td>Describe supporting content</td>
</tr>
<tr>
<td>Describe supporting functions</td>
</tr>
<tr>
<td>Describe non-functional requirements</td>
</tr>
<tr>
<td>Describe the context space</td>
</tr>
</tbody>
</table>

| 8. Develop metaphor                           |
|     Describe base and overlay metaphors        |
|     Find metaphor integration                 |
|     Develop templates for deployment          |

<table>
<thead>
<tr>
<th>Precondition</th>
<th>Contracted collaboration of all partners</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Real necessity for WIS development</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Postcondition</th>
<th>Description of application domain accepted and consistent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>New business rule model accepted</td>
</tr>
</tbody>
</table>

Information analysis is based on WIS storyboard pragmatics. We usually start with templates that specify the brand of the WIS by the four dimensions provider, content, receiver, and main actions. The brand is based on the mission containing general statements on the content, the user, the tasks, purposes, and benefits. The content indication may be extended on the basis of the content item specification frame and by a rough description of context based on the general context template. The brand description is compared with the description of tasks to be supported. Tasks are specified through the general characterisation template. The specification of the intention of the WIS is based on the designation of goals of the WIS.

Additionally we integrate metaphors. They must be captured as early as possible. Metaphors may be based on the style guides that are provided by the customer. Metaphorical structures for WIS can be developed based on linguistic and cognitive research [4].

The application domain description feeds the derivation of the WIS utilisation portfolio. The general section of the WIS utilisation portfolio consists of a list of categories, refined brands, and a general description of kinds of WIS portfolio. The partner section of the WIS utilisation portfolio surveys partners, i.e. a list of actors with portfolio, possibly with rights, obligations, and roles. The storyboard section of the WIS utilisation portfolio describes scenarios used and activities which are based on a list of word fields characterising activities. Activities are characterised by an activity style, by activity pattern, and by collaboration styles and pattern. The content section of the WIS utilisation portfolio describes the content chunks together with the content portfolio, i.e. content demand, consumption, and production. The functionality section of the WIS utilisation portfolio describes supporting functions on the basis of function chunks and the functionality portfolio, i.e. demand, consumption, production of functionality. The WIS utilisation portfolio is enhanced by a designation of non-functional requirements, especially quality requirements and by a derivation of specifics of the context space.

4 Conclusions
Web information systems engineering inherits the achievements and approaches of information systems engineering. They differ in architectures that are used, in integration of suites of (information) systems, in demands for quality, in stories of utilisation etc. Therefore, WIS engineering must cope with novel requirements and lead to novel solutions.

The paper is one of the first contributions to explicit treatment of application domain descriptions for WIS. They become crucial for web information systems due to the general purpose use of websites by a large variety of users who use a large number of display clients who have very different profiles, goals, and tasks and who vary in their behaviour.

WIS engineering must be a process that is based on a well-defined methodology that is manageable. We applied a SPICE evaluation to our first methodology and improved this methodology based on the critics and findings. This paper discusses the improvement process and illustrates the results for the first engineering process.

Software process improvement applied to methodology development has potential, but is also demanding due to the fact that methodologies are more abstract than e.g. software projects. For instance, evidence in assessments can be interpreted quite freely. In this case, the main benefits were found in improving the structure of the methodology presentation.

WIS engineering is based on very early and stable integration of business users. These users do not want to control code or documents but are mainly interested in the result. Therefore, each specification step must lead to an executable specification.

References


