

Dines Bjorner: 9th DRAFT: October 31, 2008

3.1 Discussion of The Requirements Concept

3.1.1 Some Principles

The objective of requirements engineering is to create a requirements prescription.

IEEE Definition of ‘Requirements’

By a requirements we understand (cf. IEEE Standard 610.12 [102]): *“A condition or capability needed by a user to solve a problem or achieve an objective”*.

“slide 493”

The above definition¹ is adequate for our purposes. It stresses what requirements are. It is not operational, and that is good. It does not define the thing, the requirements, by how they look, or how you construct them. The ‘what’ (not the ‘how’) of requirements is the purpose of this and the next seven chapters.

“slide 494”

The “Golden Rule” of Requirements Engineering

Principle 5 (Requirements Engineering [1])]Prescribe only those requirements that can be objectively shown to hold for the designed software. ■

“Objectively shown” means that the designed software can either be proved (verified), or be model checked, or be tested, to satisfy the requirements.

“slide 495”

An “Ideal Rule” of Requirements Engineering

Principle 6 (Requirements Engineering [2])]When prescribing requirements, formulate, at the same time, tests (theorems, properties for model checking) whose actualisation should show adherence to the requirements. ■

¹ We shall mostly be using the term ‘requirements’ in its plural form, but think of it as “one body” of such!

The rule is labelled “ideal”. We shall not show such precautions in this volume. They ought to be shown. But either we would show one, or a few instances, and they would “drown” in the mass of material otherwise presented. Or they would, we claim, trivially take up too much space. The rule is clear. It is a question for proper management to see that it is adhered to.

Principle 7 (Requirements Adequacy) Make sure that requirements cover what users expect. ■

That is, do not express a requirement for which you have no users, but make sure that all users’ requirements are represented or somehow accommodated. In other words: the requirements gathering process needs to be like an extremely “fine-meshed net”: One must make sure that all possible stakeholders have been involved in the requirements acquisition process, and that possible conflicts and other inconsistencies have been obviated.

The approach which we put forward in this chapter, namely “deriving” domain requirements “directly” from domain descriptions, and systematically conceiving interface requirements also from domain descriptions shall help secure adherence to this ‘Requirements Adequacy’ principle. We shall soon explain the terms ‘domain requirements’ and ‘interface requirements’.

Principle 8 (Requirements Implementability) Make sure that requirements are implementable. ■

That is, do not express a requirement for which you have no assurance that it can be implemented. In other words, although the requirements phase is not a design phase, one must tacitly assume, perhaps even indicate, somehow, that an implementation is possible. But the requirements in and by themselves, stay short of expressing such designs.

Principle 9 (Requirements Verifiability and Validability) Make sure that requirements are verifiable and can be validated. ■

That is, do not express a requirement for which you have no assurance that it can be verified and validated. In other words, once a first-level software design has been proposed, one must show that it satisfies the requirements. Thus specific parts of even abstract software designs are usually provided with references to specific parts of the requirements that they are (thus) claimed to implement.

We conclude this discussion.

Characterisation 71 (Requirements) By *requirements* we shall understand a document which prescribes desired properties of a machine: (i) what entities the machine shall “maintain”, and what the machine shall (must; not should) offer of (ii) functions and of (iii) behaviours (iv) while also expressing which events the machine shall “handle”. ■

3.1.2 One Domain, Many Requirements

“SLIDE 501”

Domain descriptions nearly always cover a much larger span than do any one individual (set of) requirements, that is, a requirements for a specific software product. Thus one can expect one domain description to be the basis for many (more-or-less) distinct requirements prescriptions, which, since they are (to be) based on the same domain description, and when they share some domain phenomena and concepts, must (somehow) be made to fit one another. We shall cover the notion of requirements fitting in Sect. 3.9.3 (starting Page 131) when covering the topic of domain requirements.

3.1.3 The Machine as Target

“SLIDE 502”

A requirements prescription specifies externally observable properties of simple entities, functions, events and behaviours of **the machine** such as the requirements stakeholders wish them to be.

3.1.4 Machine = Hardware + Software

“SLIDE 503”

The **machine** is what is required, that is, the **hardware** and **software** that is to be designed and which are to satisfy the requirements.

3.1.5 On “Derivation” of Requirements

“SLIDE 504”

It is a highlight of this book that requirements engineering has a scientific foundation and that that scientific foundation is the domain theory, that is the properties of the domain as modelled by a domain description. Conventional requirements engineering, as covered in a great number of software engineering textbooks [159, 165, 183, 203, 78], does not have (such) a scientific foundation. This foundation allows us to pursue requirements engineering in quite a new manner.

“slide 505”

The way in which we shall pursue the central core of requirements engineering will now be sketched. When modelling the requirements, after all the initial stages of requirements stakeholder identification, etc., we “divide” the work into three kinds of requirements: (i) the domain requirements, (ii) the interface requirements, and (iii) the machine requirements.

“slide 506”

These sub-stages are related to their underlying domain as follows. (i) The domain requirements are those requirements which can be expressed solely using terms from the domain (in addition to ordinary, say, English. (iii) The machine requirements are those requirements which can be expressed solely using terms from the machine, that is, the hardware and software regime. (ii) The interface requirements are then those requirements which can be expressed using terms from both the domain and the machine.

“slide 507”

This approach materially alters the way in which we pursue the preparatory requirements engineering stages of stakeholder identification, requirements acquisition, requirements analysis and concept formation, business process re-engineering and requirements terminology.

We will cover these preparatory requirements engineering stages in two rounds: in an overview fashion, Sect. 3.2.1, and in some detail, Sects. 3.3–3.8. In our coverage we rely on the reader having first carefully studied the “similarly named” domain engineering stage.

3.1.6 Summary

“SLIDE 508”

A **requirements prescription** thus (**putatively**) expresses what there should be. A requirements prescription expresses nothing about the design of the possibly desired (required) software.

• • •

We shall show how a major part of a requirements prescription can be “derived” from “its” prerequisite domain description.

3.2 Stages of Requirements Engineering

“SLIDE 509”

3.2.1 An Overview of “What To Do?”

The present section’s main material, Pages 112–116, on “what to do” in requirements development is very much like section on “what to do” in domain development (Pages 52–54). So we kindly ask the reader to recall that former section.

We first summarise what is to be done, ending that summary with an overview listing (Page 116) (as we did for the stages of domain engineering (Page 55) . Then, in subsequent section we cover, in some detail, how to do that which is to be done (Sect. 3.3–3.13).

[1] Requirements Information²

This (numbered [1]) section is very much like the similarly named subsection ([1]) on Page 52.

The purpose of this stage of development, to repeat, is to record all relevant administrative, socio-economic, budgetary, project management (planning) and such non-formalisable information which has a bearing on the requirements prescription project.

For more specifics on this topic we refer to Sect. 3.3 (Page 117).

For the example ‘requirements information’ document we refer to (Appendix) Sect. M.1, Pages 365–370.

² “SLIDE 510”

[2] Requirements Stakeholder Identification³

This (numbered [2]) section is very much like the similarly named subsection ([2]) on Page 52.

The purpose of this stage of development is to identify the requirements stakeholders. They are usually a proper subset of the domain stakeholders. One does not need to interact with all domain stakeholders as that was supposedly done in the domain description development phase, and the domain description is, of course, assumed available and the basis for the requirements prescription development phase. For more specifics on this topic we refer to Sect. 3.4

For the example ‘requirements stakeholder identification’ document we refer to (Appendix) Sect. M.2, Pages 370–370.

[3] Requirements Acquisition⁴

This (numbered [3]) subsection is very much like the similarly named subsection ([3]) on Page 53.

The purpose of this stage of development is to acquire and elicit, to list, to enumerate, to collect the requirements. Now since, as we shall see, the requirements shall consist of three parts: the domain requirements, the interface requirements and the machine requirements, and since the first two relies very strongly on the domain description, a major part of the requirements acquisition takes on a rather different form than the way in which domain acquisition takes place.

For more specifics on this topic we refer to Sect. 3.5

For the example ‘requirements acquisition’ document we refer to (Appendix) Sect. M.3, Pages 370–370.

[4] Requirements Analysis & Concept Formation⁵

This (numbered [4]) subsection is very much like the similarly named subsection ([4]) on Page 53.

The purpose of this stage of development is to analyse and conceive of concepts around which to structure the requirements, for those relevant such concepts that are not already part of the domain description.

While performing the steps of (a) the domain requirements, (b) the interface requirements and (c) the machine requirements stages of development these steps may give rise to inconsistencies incompletenesses for which analysis has to check their absence.

While deriving the above three stages (a–b–c) analyses must be made to secure that the required simple entities, functions, events and behaviours are

³ “SLIDE 511”

⁴ “SLIDE 512”

⁵ “SLIDE 513”

computable. (For a domain description incomputable concepts are allowed.) (In fact, the whole purpose of requirements construction is to secure some form of computability⁶.

For more specifics on this topic we refer to Sect. 3.6

For the example ‘requirements analysis’ document we refer to (Appendix) Sect. M.4, Pages 371–371.

[5] Requirements Business Process Re-Engineering⁷

This (numbered [5]) subsection is very much like the similarly named subsection ([5]) on Page 53.

The background for this stage of development is the set of current business processes as carried out in the domain at present and as described during construction of the domain description. The purpose of this stage of development is to prescribe how the business processes are to be in future, once the required software has been installed; that is: the business processes often need be re-engineered since that is (to be) assumed by the required software.

For more specifics on this topic we refer to Sect. 3.7

For the example ‘business process re-engineering’ document we refer to (Appendix) Sect. M.5, Pages 371–373.

[6] Requirements Terminology⁸

This (numbered [6]) subsection is very much like the similarly named subsection ([6]) on Page 53.

The purpose of this stage of development, one which “links” up to the domain terminology, is to extend that domain terminology with the new terms that arise as a consequence of interface and machine requirements. (The stages of development [interface and machine requirements] will be covered later.)

For more specifics on this topic we refer to Sect. 3.8

For the example ‘requirements terminology’ document we refer to (Appendix) Sect. M.6, Pages 373–373.

[7] Requirements Modelling⁹

This (numbered [7]) subsection is very much like the similarly named subsection ([7]) on Page 54.

⁶ By this “some form” is meant: Either the concepts are computable or, through interaction with an environment (a human or otherwise) the desired effect can be achieved.

⁷ “SLIDE 515”

⁸ “SLIDE 516”

⁹ “SLIDE 517”

The purpose of this stage of development is to develop a requirements prescription, that is a narrative and a formal specification of three requirements facets: domain requirements, interface requirements and machine requirements.

“slide 518”

These relate as follows: domain requirements are those requirements which can be expressed solely by using terms from the domain (and otherwise ordinary terms of English); machine requirements are those requirements which can be expressed solely by using terms “from” the machine, i.e., hardware and software terms (and otherwise ordinary terms of English); interface requirements are those requirements which can only be expressed using terms both of the domain and of the machine.

“slide 519”

For more specifics on this topic we refer to Sect. 3.9

For the example ‘requirements prescription’ document we refer to Appendices N–P, Pages 375–393.

[8] Requirements Verification¹⁰

This (numbered [8]) subsection is very much like the similarly named subsection ([8]) on Page 54.

The purpose of this stage of development, which normally goes hand-in-hand with requirements modelling, is to verify (prove, model check and/or test) properties of what is being prescribed. Examples of requirements prescription verification are that all arguments to defined functions are within their range of applicability, that postulated functions can be implemented, that defined functions are of a desired complexity, etcetera.

For more specifics on this topic we refer to Sect. 3.10

For the example ‘requirements verification’ document we refer to (Appendix) Sect. Q.1, Pages 395–395.

[9] Requirements Validation¹¹

This (numbered [9]) subsection is very much like the similarly named subsection ([9]) on Page 54.

The purpose of this stage of development which normally comes after proper completion of a requirements document, is to make sure that what has been prescribed is actually what the relevant requirements stakeholder have requested. The validation process is necessarily informal in that it relies solely on the narrative prescriptions as these are the only ones that all requirements stakeholders understand.

For more specifics on this topic we refer to Sect. 3.11

For the example ‘requirements validation’ document we refer to (Appendix) Sect. Q.2 Pages 395–395.

¹⁰ “SLIDE 520”

¹¹ “SLIDE 521”

[10] Requirements Satisfiability and Feasibility¹²

This (numbered [10]) section is new. It has no direct counterpart in domain engineering.

The purpose of this stage of development is to secure that the full requirements are implementable and in a way that is economic and relevant. By feasible (i.e., relevance) we mean that an implementation does not consume unreasonable resources: time, space, etcetera.

For more specifics on this topic we refer to Sect. 3.12

For the example ‘requirements satisfiability and feasibility’ document we refer to (Appendix) Sect. Q.3 Pages, 395–395.

[11] Requirements Theory Formation¹³

This (numbered [11]) subsection is very much like the similarly named subsection ([11]) on Page 54.

The purpose of this stage of development further develop the domain theory, Sect. 2.13, by examining the theorems of that theory as to their also holding for the specific requirements hold and to develop such theorems that hold for the specific requirements.

For more specifics on this topic we refer to Sect. 3.13

For the example ‘requirements theory formation’ document we refer to (Appendix) Sect. Q.4 Pages 395–395.

3.2.2 A Summary Enumeration

“SLIDE 524”

1 Requirements Information	Sect. 3.3 (Page 117)
2 Requirements Stakeholder Identification	Sect. 3.4 (Page 119)
3 Requirements Acquisition	Sect. 3.5 (Page 119)
4 Requirements Analysis & Concept Formation	Sect. 3.6 (Page 121)
5 Business Process Re-Engineering	Sect. 3.7 (Page 121)
6 Requirements Terminology	Sect. 3.8 (Page 127)
7 Requirements Modelling	Sect. 3.9 (Page 127)
(a) Domain Requirements	Sect. 3.9.3 (Page 128)
(b) Interface Requirements	Sect. 3.9.4 (Page 133)
(c) Machine Requirements	Sect. 3.9.5 (Page 135)
8 Requirements Verification	Sect. 3.10 (Page 137)
9 Requirements Validation	Sect. 3.11 (Page 137)
10 Requirements Satisfiability and Feasibility	Sect. 3.12 (Page 137)
11 Requirements Theory Formation	Sect. 3.13 (Page 137)

¹² “SLIDE 522”

¹³ “SLIDE 523”

3.3 Requirements Information

“SLIDE 525”

This (numbered [1]) section is very much like the similarly named subsection ([1]) on Page 52.

For the example ‘requirements information’ document we refer to (Appendix) Sect. M.1, Pages 365–370.

We have earlier, as mentioned above, extensively (Pages 6–24) covered the general issues of informative documents. The reader is strongly encouraged to review those pages, Sect. 1.5. Suffice it here to emphasize the following.

We highlight, in the next many “highlighted” paragraphs, what is special, to requirements prescription developments, with respect to the many items of project information.

Current Situation: Cf. Sect. 1.6.3, Page 9 “slide 526”

As mentioned in Sect. 1.6.3 on page 9 the context in which the requirements developments starts must be emphasized. That context invariably includes the existence of a domain description. Please no reference to possible software designs.

We refer to Appendix Sect. M.1.3 starting on Page 366 for an example related to the *current situation* topic.

Needs and Ideas: Cf. Sect. 1.6.4, Pages 9–10 “slide 527”

TO BE WRITTEN

We refer to Appendix Sect. M.1.4 starting on Page 366 for an example related to the *needs and ideas* topic.

Concepts and Facilities: Cf. Sect. 1.6.5, Pages 10–11 “slide 528”

TO BE WRITTEN

We refer to Appendix Sect. M.1.5 starting on Page 367 for an example related to the *concepts and facilities* topic.

Scope and Span: Cf. Sect. 1.6.6, Page 11 “slide 529”

TO BE WRITTEN

We refer to Appendix Sect. M.1.6 starting on Page 368 for an example related to the *scope and span* topic.

Assumptions and Dependencies: Cf. Sect. 1.6.7, Pages 11–12 “slide 530”

TO BE WRITTEN

We refer to Appendix Sect. M.1.7 starting on Page 369 for an example related to the *assumptions and dependencies* topic.

Implicit/Derivative Goals: Cf. Sect. 1.6.8, Page 12 “slide 531”

TO BE WRITTEN

We refer to Appendix Sect. M.1.8 starting on Page 369 for an example related to the *implicit/derivative goals* topic.

Synopsis: Cf. Sect. 1.6.9, Page 13 “slide 532”

TO BE WRITTEN

We refer to Appendix Sect. M.1.10 starting on Page 369 for an example related to the *synopsis* topic.

“slide 533”

Software Development Graphs: Cf. Sect. 1.6.10, Pages 13–15

TO BE WRITTEN

We refer to Appendix Sect. M.1.11 starting on Page 369 for an example related to the *software development graph* topic.

“slide 534”

Resource Allocation: Cf. Sect. 1.6.11, Pages 15–16

TO BE WRITTEN

We refer to Appendix Sect. M.1.12 starting on Page 369 for an example related to the *resource allocation* topic.

“slide 535”

Budget (and Other) Estimates: Cf. Sect. 1.6.12, Page 16

TO BE WRITTEN

We refer to Appendix Sect. M.1.13 starting on Page 370 for an example related to the *budget (and other) estimates* topic.

“slide 536”

Standards Compliance: Cf. Sect. 1.6.13, Pages 16–19

TO BE WRITTEN

We refer to Appendix Sect. M.1.14 starting on Page 370 for an example related to the *standards compliance* topic.

“slide 537”

Contracts and Design Briefs: Cf. Sect. 1.6.14, Pages 19–23

TO BE WRITTEN

We refer to Appendix Sect. M.1.15 starting on Page 370 for an example related to the *contract and design brief* topic.

“slide 538”

MORE TO COME

3.4 Requirements Stakeholders

“SLIDE 539”

This section is an expansion on Item [2] on Page 52 and Sect. 2.4

On Page 113 we wrote that the set of requirements stakeholder *is usual* a proper subset of of the domain stakeholders. We may need to modify this statement. We assume that a list of domain stakeholders omitted administrative staff from the areas of general services staff: accounting, archiving (journal), general procurement, etcetera. We assume they were omitted since the domain, “technically speaking” was not about these areas. The domain, for example, may be about railways for which we may have included passenger ticketing staff but for which we omitted for example the “back room” accounting staff. That latter staff would, in principle, not know whether they were doing accounting for a railway-related company or for a financial service system or for a hospital. If the requirements are for such general services of a domain which do indeed impinge on such, previously omitted staff, then such staff group must be included among the stakeholders. (But we do remind the reader that our domain is in such a case extended with the phenomena and concepts of the areas of general services thus identified — and appropriate domain extensions must be provided for.)

“slide 540”

For the example ‘requirements stakeholder identification’ document we refer to (Appendix) Sect. M.2, Pages 370–370. We refer to Appendix Sect. M.2 starting on Page 370 for an example related to the *requirements stakeholder* topic.

3.5 Requirements Acquisition

“SLIDE 541”

We have, in Sect. 3.2.1, Item [3], on Page 113, outlined the three stages of requirements modelling: domain requirements, interface requirements and machine requirements. These requirements modelling stages were first overviewed, in more detail in Sect. 3.1.5, Pages 111–111. The three requirements modelling stages will be dealt with in “final” detail in Sects. 3.9.3–3.9.5, Pages 128–137.

We shall assume you have followed those brief outlines. Requirements acquisition now basically follows these three kinds of requirements modelling stages.

3.5.1 Domain Requirements Acquisition

“SLIDE 542”

Domain requirements acquisition is based solely on the domain description. From that domain description is developed, in sub-stages (or steps) of development a domain requirements acquisition document. The steps are *projection*, *instantiation*, *determination*, *extension* and *fitting*. Each step takes a document and results in a requirements acquisition document. The first step takes a domain description document. All steps results in a domain requirements acquisition document.

“slide 543”

Sect. 3.9.3 shall outline the details of the *projection*, *instantiation*, *determination*, *extension* and *fitting* operations. In the domain requirements acquisition stage one basically marks up a “requirements acquisition copy” of the domain description. In the domain requirements (Sect. 3.9.3) acquisition stage one collects the ‘domain requirements acquisition’ mark-ups into a consistent acquisition document editing it as per the mark-ups. All domain-to-domain requirements acquisition operations (*projection*, *instantiation*, *determination*, *extension* and *fitting*) are conducted interactively, between the domain engineer and all the relevant requirement stakeholder, in turn, one after the other, and, in principle, in no particular order. The domain engineer, in a sense, guides the requirement stakeholders through the emerging domain requirements acquisition prescription document determining what to *project*, *instantiate*, make more *deterministic*, *extend* and *fit*.

3.5.2 Interface Requirements Acquisition

Interface requirements acquisition is based on the domain requirements acquisition document and on some awareness of the facilities of machine being required. Awareness and determination of which these facilities are, or could be, evolve as a result of conducting the below steps of interface requirements. First the domain engineer and the relevant requirement stakeholders determine which phenomena simple entities, functions, events and behaviours of the domain need be represented by the machine. Then they go through each of these kinds of phenomena to determine what sharing means: what properties of simple entities, functions, events and behaviours are to be satisfied by the machine. This entails decisions on abstractions of initial data input and refreshment, interactive computation of functions between the machine and the domain, “translation” of events in the domain into the machine, and interactive behaviours between the machine and the domain. Again, each of these steps result in the input requirements acquisition document being further annotated — as were the domain requirements acquisition documents. Proper requirements modelling documents now take these annotated acquisition documents and rework them, “clean them up”, into proper requirements prescription documents.

3.5.3 Machine Requirements Acquisition

“SLIDE 547”

Machine requirements acquisition is basically concerned with the acquisition of such required properties of the desired machine as machine: performance, dependability, maintenance, platform and documentation. We shall defer the acquisition aspects of these machine requirements till we later, Sect. 3.9.5, treat machine requirements modelling.

• • •

We refer to Appendix Sect. M.3 starting on Page 370 for an example related to the *requirements acquisition* topic.

3.6 Analysis and Concept Formation

“SLIDE 548”

TO BE WRITTEN

We refer to Appendix Sect. M.4 starting on Page 371 for an example related to the *requirements analysis and concept formation* topic.

3.7 Business Process Re-Engineering

“SLIDE 549”

Characterisation 72 (Business Process Reengineering) By *business process reengineering* we understand the reformulation of previously adopted business process descriptions, together with additional business process engineering work. ■

3.7.1 What Are BPR Requirements?

“SLIDE 550”

Two “paths” lead to business process reengineering:

- A client wishes to improve enterprise operations by deploying new computing systems (i.e., new software). In the course of formulating requirements for this new computing system a need arises to also reengineer the human operations within and without the enterprise.
- An enterprise wishes to improve operations by redesigning the way staff operates within the enterprise and the way in which customers and staff operate across the enterprise-to-environment interface. In the course of formulating reengineering directives a need arises to also deploy new software, for which requirements therefore have to be enunciated.

“slide 551”

One way or the other, business process reengineering is an integral component in deploying new computing systems.

3.7.2 Overview of BPR Operations

“SLIDE 552”

We suggest six domain-to-business process reengineering operations:

- 1 introduction of some new and removal of some old *intrinsic*s;
- 2 introduction of some new and removal of some old *support technologies*;
- 3 introduction of some new and removal of some old *management and organisation substructures*;
- 4 introduction of some new and removal of some old *rules and regulations*;
- 5 introduction of some new and removal of some old work practices (relating to *human behaviours*); and
- 6 related *scripting*.

3.7.3 BPR and the Requirements Document

“SLIDE 553”

Requirements for New Business Processes

The reader must be duly “warned”: The BPR requirements are not for a computing system, but for the people who “surround” that (future) system. The BPR requirements state, unequivocally, how those people are to act, i.e., to use that system properly. Any implications, by the BPR requirements, as to concepts and facilities of the new computing system must be prescribed (also) in the domain and interface requirements.

Place in Narrative Document¹⁴

We shall thus, in Sects. 3.7.4–3.7.8, treat a number of BPR facets. Each of whatever you decide to focus on, in any one requirements development, must be prescribed. And the prescription must be put into the overall requirements prescription document.

As the BPR requirements “rebuilds” the business process description part of the domain description¹⁵, and as the BPR requirements are not directly requirements for the machine, we find that they (the BPR requirements texts) can be simply put in a separate section.

There are basically two ways of “rebuilding” the domain description’s business process’s description part (D_{BP}) into the requirements prescription part’s BPR requirements (R_{BPR}). Either you keep all of D as a base part in R_{BPR} , and then you follow that part (i.e., R_{BPR}) with statements, R'_{BPR} , that express the new business process’s “differences” with respect to the “old” (D_{BP}). Call the result R_{BPR} . Or you simply rewrite (in a sense, the whole of) D_{BP} directly into R_{BPR} , copying all of D_{BP} , and editing wherever necessary.

Place in Formalisation Document¹⁶

The above statements as how to express the “merging” of BPR requirements into the overall requirements document apply to the narrative as well as to the formalised prescriptions.

We may assume that there is a formal domain description, \mathcal{D}_{BP} , (of business processes) from which we develop the formal prescription of the BPR requirements. We may then decide to either develop entirely new descriptions of the new business processes, i.e., actually prescriptions for the business reengineered processes, \mathcal{R}_{BPR} ; or develop, from \mathcal{D}_{BP} , using a suitable schema calculus, such as the one in RSL, the requirements prescription \mathcal{R}_{BPR} , by suitable parameterisation, extension, hiding, etc., of the domain description \mathcal{D}_{BP} .

¹⁴ “SLIDE 554”

¹⁵ — Even if that business process description part of the domain description is “empty” or nearly so!

¹⁶ “SLIDE 557”

3.7.4 Intrinsic Review and Replacement

“SLIDE 559”

Characterisation 73 (Intrinsic Review and Replacement) By *intrinsic review and replacement* we understand an evaluation as to whether current intrinsic stays or goes, and as to whether newer intrinsic need to be introduced. ■

“slide 560”

Example. 1 – Intrinsic Replacement: *A railway net owner changes its business from owning, operating and maintaining railway nets (lines, stations and signals) to operating trains. Hence the more detailed state changing notions of rail units need no longer be part of that new company’s intrinsic while the notions of trains and passengers need be introduced as relevant intrinsic.*

•

Replacement of intrinsic usually point to dramatic changes of the business and are usually not done in connection with subsequent and related software requirements development.

3.7.5 Support Technology Review and Replacement

“SLIDE 561”

Characterisation 74 (Support Technology Review and Replacement) By *support technology review and replacement* we understand an evaluation as to whether current support technology as used in the enterprise is adequate, and as to whether other (newer) support technology can better perform the desired services. ■

“slide 562”

Example. 2 – Support Technology Review and Replacement: *Currently the main information flow of an enterprise is taken care of by printed paper, copying machines and physical distribution. All such documents, whether originals (masters), copies, or annotated versions of originals or copies, are subject to confidentiality. As part of a computerised system for handling the future information flow, it is specified, by some domain requirements, that document confidentiality is to be taken care of by encryption, public and private keys, and digital signatures. However, it is realised that there can be a need for taking physical, not just electronic, copies of documents. The following business process reengineering proposal is therefore considered: Specially made printing paper and printing and copying machines are to be procured, and so are printers and copiers whose use requires the insertion of special signature cards which, when used, check that the person printing or copying is the person identified on the card, and that that person may print the desired document. All copiers will refuse to copy such copied documents — hence the special paper. Such paper copies can thus be read at, but not carried outside the premises (of the printers and copiers). And such printers and copiers can register who printed, respectively who tried to copy, which documents. Thus people are now responsible for the security (whereabouts) of possible paper*

“slide 563”

copies (not the required computing system). The above, somewhat construed example, shows the “division of labour” between the contemplated (required, desired) computing system (the “machine”) and the “business reengineered” persons authorised to print and possess confidential documents. “slide 564”

It is implied in the above that the reengineered handling of documents would not be feasible without proper computing support. Thus there is a “spill-off” from the business reengineered world to the world of computing systems requirements. ●

3.7.6 Management and Organisation Reengineering “SLIDE 565”

Characterisation 75 (Management and Organisation Reengineering)

By *management and organisation reengineering* we understand an evaluation as to whether current management principles and organisation structures as used in the enterprise are adequate, and as to whether other management principles and organisation structures can better monitor and control the enterprise. ■

Example. 3 – Management and Organisation Reengineering: *A rather complete computerisation of the procurement practices of a company is being contemplated. Previously procurement was manifested in the following physically separate as well as designwise differently formatted paper documents: requisition form, order form, purchase order, delivery inspection form, rejection and return form, and payment form. The supplier had corresponding forms: order acceptance and quotation form, delivery form, return acceptance form, invoice form, return verification form, and payment acceptance form. The current concern is only the procurement forms, not the supplier forms.*

The proposed domain requirements are mandating that all procurer forms disappear in their paper version, that basically only one, the procurement document, represents all phases of procurement, and that order, rejection and return notification slips, and payment authorisation notes, be effected by electronically communicated and duly digitally signed messages that represent appropriate subparts of the one, now electronic procurement document.

The business process reengineering part may now “short-circuit” previous staff’s review and acceptance/rejection of former forms, in favour of fewer staff interventions.

The new business procedures, in this case, subsequently find their way into proper domain requirements: those that support, that is monitor and control all stages of the reengineered procurement process. ●

3.7.7 Rules and Regulations Reengineering “SLIDE 569”

Characterisation 76 (Rules and Regulation Reengineering) By *rules and regulations reengineering* we understand an evaluation as to whether

current rules and regulations as used in the enterprise are adequate, and as to whether other rules and regulations can better guide and regulate the enterprise. ■

Here it should be remembered that rules and regulations principally stipulate business engineering processes. That is, they are — i.e., were — usually not computerised. “slide 570”

Example. 4 – Rules and Regulations Reengineering: *Assume now, due to reengineered support technologies, that interlock signalling can be made magnitudes safer than before, without interlocking. Thence it makes sense to reengineer a domain rule of from: In any three-minute interval at most one train may either arrive to or depart from a railway station into: In any 20-second interval at most two trains may either arrive to or depart from a railway station.*

This reengineered rule is subsequently made into a domain requirements, namely that the software system for interlocking is bound by that rule. ●

3.7.8 Script Reengineering “SLIDE 571”

On one hand, there is the engineering of the contents of rules and regulations, and, on another hand, there are the people (management, staff) who script these rules and regulations, and the way in which these rules and regulations are communicated to managers and staff concerned. “slide 572”

Characterisation 77 (Script Reengineering) *By script reengineering we understand evaluation as to whether the way in which rules and regulations are scripted and made known (i.e., posted) to stakeholders in and of the enterprise is adequate, and as to whether other ways of scripting and posting are more suitable for the enterprise. ■* “slide 573”

MORE TO COME

“slide 574”

MORE TO COME

“slide 575”

MORE TO COME

3.7.9 Human Behaviour Reengineering

“SLIDE 576”

Characterisation 78 (Human Behaviour Reengineering) By *human behaviour reengineering* we understand an evaluation as to whether current human behaviour as experienced in the enterprise is acceptable, and as to whether partially changed human behaviours are more suitable for the enterprise. ■

“slide 577”

Example. 5 – Human Behaviour Reengineering: *A company has experienced certain lax attitudes among members of a certain category of staff. The progress of certain work procedures therefore is reengineered, implying that members of another category of staff are henceforth expected to follow up on the progress of “that” work.*

In a subsequent domain requirements stage the above reengineering leads to a number of requirements for computerised monitoring of the two groups of staff. ●

3.7.10 Discussion: Business Process Reengineering

“SLIDE 578”

Who Should Do the Business Process Reengineering?

It is not in our power, as software engineers, to make the kind of business process reengineering decisions implied above. Rather it is, perhaps, more the prerogative of appropriately educated, trained and skilled (i.e., gifted) other kinds of engineers or business people to make the kinds of decisions implied above. Once the BP reengineering has been made, it then behooves the client stakeholders to further decide whether the BP reengineering shall imply some requirements, or not.

Who Should Do the Business Process Reengineering?¹⁷

Once that last decision has been made in the affirmative, we, as software engineers, can then apply our abstraction and modelling skills, and, while collaborating with the former kinds of professionals, make the appropriate prescriptions for the BPR requirements. These will typically be in the form of domain requirements, which are covered extensively in Sect. ??.

General¹⁸

Business process reengineering is based on the premise that corporations must change their way of operating, and, hence, must “reinvent” themselves. Some corporations (enterprises, businesses, etc.) are “vertically” structured along

¹⁷ “SLIDE 579”

¹⁸ “SLIDE 580”

functions, products or geographical regions. This often means that business processes “cut across” vertical units. Others are “horizontally” structured along coherent business processes. This often means that business processes “cut across” functions, products or geographical regions. In either case adjustments may need to be made as the business (i.e., products, sales, markets, etc.) changes.

We refer to Appendix Sect. M.5 starting on Page 371 for an example related to the *business process re-engineering* topic.

3.8 Requirements Terminology

“SLIDE 581”

TO BE WRITTEN

We refer to Appendix Sect. M.6 starting on Page 373 for an example related to the *requirements terminology* topic.

“slide 582”

“slide 583”

3.9 Requirements Modelling

“SLIDE 584”

3.9.1 Aims & Objectives

The **aims** of the requirements modelling stage of requirements engineering are to finalise a **delineation** of the span for the requirements, to ensure that the requirements **relate strongly** to the domain, and to help **secure** that that the client gets the **right software**. The **objectives** of the requirements modelling stage of requirements engineering are to **develop** a proper requirements prescription, one that is well **analysed** (“theoretised”), and one that leads to **efficient** and **correct** software.

3.9.2 Requirements Facets

“SLIDE 585”

It has been highlighted, significantly, above that there are three facets to requirements modelling: modelling those, the **domain** requirements which can be expressed solely using terms from the domain, modelling those, the **interface** requirements which can be expressed using terms from both the domain and the machine, and modelling those, the **machine** requirements which can be expressed solely using terms from the machine. For interface requirements, by “*using terms both from the domain and the machine*” we mean: that the “smallest” requirements prescription units contain both kinds, that is, they are used in an inseparable way.

We refer to Appendices N–P (Pages 375–393) for an example related to the *requirements modelling* topic.

3.9.3 Domain Requirements

“SLIDE 586”

A domain requirements prescription is that part of the overall requirements prescription which can be expressed solely using terms from the domain description. Thus to construct the domain requirements prescription all we need is collaboration with the requirements stakeholders (who, with the requirements engineers, developed the BPR) and the possibly rewritten (BPR influenced) domain description — which we shall refer to as a domain requirements document.

The domain span is usually much, or just, larger than the domain implied by the requirements one is about to construct. Therefore a first domain-to-requirements operation is to **project** onto the first step domain requirements only those phenomena and concepts of the domain which are required. The resulting domain requirements projection often defines a much-too-general set of phenomena and concepts which, although they are needed, may not be needed in the generality given, hence they are **instantiated** to more specific ones. The resulting domain requirements projection & instantiation often defines its phenomena and concepts too non-deterministically, hence “excess” non-determinism is more more **determinate**. Often computing and communication allows for phenomena and concepts which are not feasible in the domain, hence we **extend** the emerging domain requirements projection & instantiation & determination with such domain-like phenomena and concepts which are feasible. Finally, requirements engineering may take place in a context where more than one group is developing requirements, but for “more-or-less” distinct “areas”, hence we suggest to **fit** the various emerging domain requirements into a consistent and coherent set of such.

Thus we end up with the following domain-to-requirements steps:

- | | |
|---|------------------|
| 1 domain-to-requirements projection | starts Page 128, |
| 2 domain-to-requirements instantiation | starts Page 129, |
| 3 domain-to-requirements determination | starts Page 130, |
| 4 domain-to-requirements extension | starts Page 130, |
| 5 and domain-requirements fitting | starts Page 131. |

We shall next treat each of these steps in some detail. But first we must now understand that whereas where the narratives and the formalisations of the domain description characterised to — “spoke of” — phenomena in the domain these, when projected, instantiated, etc., become concepts of the machine !

Domain Requirements Projection¹⁹

By a **domain projection** we mean *a subset of the domain description, one which leaves out all those domain phenomena and concepts entities, functions,*

¹⁹ “SLIDE 590”

events, and (thus) behaviours that the stakeholders do not wish represented by the machine. The resulting document is a **partial domain requirements prescription**.

We refer to Appendix Sect. N.1 Pages 375–376 for an example related to the *projection* topic.

*Guidelines*²⁰

Domain phenomena and concepts that are “removed” from the emerging partial domain requirements prescription may leave uses of these phenomena and concepts elsewhere in the remaining, emerging prescription undefined. Thus they rally cannot be fully removed. The requirements modeller must make a judicious decision as how to express these phenomena and concepts, usually in some further instantiated and more deterministic form as outlined next.

*Discussion*²¹

Thus projection is not an automatable operation. It is one which is carried out jointly between the appropriate stakeholders and the domain engineers where the latter edits the emerging outcome of projection.

‘Removing’ phenomena and concepts that are unwanted in the emerging partial domain requirements prescription was first indicated in a domain acquisition step where it was left as markings in a domain description document; it is now “completed” by proper deletions and editing of both narratives and formalisations.

*Discussion of Support Example*²²

Domain Requirements Instantiation²³

By **domain instantiation** we mean a *refinement of the partial domain requirements prescription, resulting from the projection step, in which the refinements aim at rendering the entities, functions, events, and (thus) behaviours of the partial domain requirements prescription more concrete, more specific*. Instantiations usually render these concepts less general. The resulting document is a **partial domain requirements prescription**.

We refer to Appendix Sect. N.2 Pages 377–380 for an example related to the *instantiation* topic.

²⁰ “SLIDE 591”

²¹ “SLIDE 592”

²² “SLIDE 593”

²³ “SLIDE 594”

*Guidelines*²⁴

*Discussion*²⁵

*Discussion of Support Example*²⁶

Domain Requirements Determination²⁷

By **domain determination** we mean a *refinement of the partial domain requirements prescription, resulting from the instantiation step, in which the refinements aim at rendering the entities, functions, events, and (thus) behaviours of the partial domain requirements prescription less non-determinate, more determinate*. Instantiations usually render these concepts less general. The resulting document is a **partial domain requirements prescription**.

We refer to Appendix Sect. N.3 Pages 380–383 for an example related to the *determination* topic.

*Guidelines*²⁸

*Discussion*²⁹

*Discussion of Support Example*³⁰

Domain Requirements Extension³¹

By domain extension we understand the *introduction of domain entities, functions, events and behaviours that were not feasible in the original domain, but for which, with computing and communication, there is the possibility of feasible implementations, and such that what is introduced become part of the emerging domain requirements prescription*. The resulting document is a **partial domain requirements prescription**.

We refer to Appendix Sect. N.4 Pages 383–384 for an example related to the *extension* topic.

²⁴ “SLIDE 595”

²⁵ “SLIDE 596”

²⁶ “SLIDE 597”

²⁷ “SLIDE 598”

²⁸ “SLIDE 599”

²⁹ “SLIDE 600”

³⁰ “SLIDE 601”

³¹ “SLIDE 602”

*Guidelines*³²

*Discussion*³³

*Discussion of Support Example*³⁴

Domain Requirements Fitting³⁵

The issue of requirements fitting arises when two or more software development projects are based on what appears to be the same domain. The problem then is to harmonise the two or more software development projects by harmonising, if not too late, their requirements developments. The result is usually a three or more documents two or more specific requirements documents and a document, perhaps more than one, consisting of requirements that are common to two or more of the specific requirements.

We refer to Appendix Sect. N.5 Pages 384–385 for an example related to the *fitting* topic.

“slide 607”

We thus assume that there are n domain requirements developments, $d_{r_1}, d_{r_2}, \dots, d_{r_n}$, being considered, and that these pertain to the same domain — and can hence be assumed covered by a same domain description.

“slide 608”

By requirements fitting we mean a *harmonisation of $n > 1$ domain requirements that have overlapping (common) not always consistent parts and which results in n ‘modified and partial domain requirements’, and m ‘common domain requirements’ that “fit into” two or more of the ‘modified and partial domain requirements’.*

“slide 609”

By a *modified and partial domain requirements* we mean a domain requirements which is short of (that is, is missing) some description parts: text and formula. By a *common domain requirements* we mean a domain requirements. By the m common domain requirements parts, *cdrs*, *fitting into* the n modified and partial domain requirements we mean that there is for each modified and partial domain requirements, $mapdr_i$, an identified subset of *cdrs* (could be all of *cdrs*), *scdrs*, such that textually conjoining *scdrs* to *mapdr* can be claimed to yield the “original” d_{r_i} .

*A Requirements Fitting Procedure*³⁶

Requirements fitting consists primarily of a pragmatically determined sequence of analytic and synthetic (‘fitting’) steps. It is first decided which n domain requirements documents to fit. Then a ‘manual’ analysis is made of the selected, n domain requirements. During this analysis tentative common domain requirements are identified. It is then decided which m common

³² “SLIDE 603”

³³ “SLIDE 604”

³⁴ “SLIDE 605”

³⁵ “SLIDE 606”

³⁶ “SLIDE 610”

domain requirements to single out. This decision results in a tentative construction of n modified and partial domain requirements. An analysis is made of the tentative modified and partial and also common domain requirements. A decision is then made whether to accept the resulting documents or to iterate the steps above.

*Requirements Fitting Verification*³⁷

Domain Requirements Consolidation³⁸

After projection, instantiation, determination, extension and fitting, it is time to review, consolidate and possibly restructure (including re-specify) the domain requirements prescription before the next stage of requirements development.

We refer to Sect. 2.9.10 Pages 101–102 (Consolidation of Domain Facets Description) for remarks similar to the below.

The many previous domain requirements stages may have yielded descriptions which, typically at the formal level, does not reveal how it all “hangs together”. In such cases, and in general, consolidation of these domain requirements documentaton stages could take the following forms.

With each potential management unit we associate a process or an indexed set of two or more processes, usually an indeterminate number. Such management units will usually involve entities and behaviours — whether staff of entity behaviours. Usually type definitions and axioms (about sorts) and value definitions of auxiliary and well-formedness functions about values can be kept separate from the process definitions. The entity processes usually take, as arguments, the entity whose time-wise behaviour and interaction with oother entity processes is being requirements modelled.

With each structural component of the organisation we associate one or more channels, or vector or array or tensor (or . . .) indexed sets of channels.

MORE TO COME

• • •

We refer to Appendix N (Pages 375–385) for an example related to the *domain requirements* topic.

“SLIDE 615”

³⁷ “SLIDE 611”

³⁸ “SLIDE 612”

3.9.4 Interface Requirements

“SLIDE 617”

By an **interface requirements** we mean a requirements prescription which refines and extends the domain requirements by considering those requirements of the domain requirements whose entities, operations, events and behaviours are **“shared”** between the domain and the machine (being requirements prescribed).

Domain/Machine Sharing³⁹

‘Sharing’ means (a) that an **entity** is represented both in the domain and “inside” the machine, and that its machine representation must at suitable times reflect its state in the domain; (b) that an **operation** requires a sequence of several “on-line” interactions between the machine (being requirements prescribed) and the domain, usually a person or another machine; (c) that an **event** arises either in the domain, that is, in the environment of the machine, or in the machine, and need be communicated to the machine, respectively to the environment; and (d) that a **behaviour** is manifested both by actions and events of the domain and by actions and events of the machine.

“slide 619”

“slide 620”

So a systematic reading of the domain requirements shall result in an identification of all shared entities, operations, events and behaviours.

“slide 621”

Each such shared phenomenon shall then be individually dealt with: **entity sharing** shall lead to interface requirements for data initialisation and refreshment; **operation sharing** shall lead to interface requirements for interactive dialogues between the machine and its environment; **event sharing** shall lead to interface requirements for how such event are communicated between the environment of the machine and the machine. **behaviour sharing** shall lead to interface requirements for action and event dialogues between the machine and its environment.

“slide 622”

• • •

We shall now illustrate these domain interface requirements development steps with respect to our ongoing example.

“slide 623”

Interface Modalities

Data Communication

“slide 624”

Digital Sampling

“slide 625”

Tactile: Keyboards &c.

“slide 626”

Visual: Displays, Lamps, &c.

“slide 627”

Audio: Voice, Alarms, &c.

“slide 628”

Other Sensory Interface Modalities

“slide 629”

“slide 630”

Entities: Domain/Machine Sharing

Data Initialisation

“slide 631”

Data Refreshment

“slide 632”

“slide 633”

Functions: Domain/Machine Sharing

Interactive Human/Machine Dialogues

“slide 634”

Interactive Machine/Machine Protocols

“slide 635”

“slide 636”

Events: Domain/Machine Sharing

Human/Machine/Human Events

“slide 637”

Machine/Machine Events

“slide 638”

Other Context/Machine Events

“slide 639”

“slide 640”

Behaviour: Domain/Machine Sharing

Human/Machine/Human Behaviours

“slide 641”

Machine/Machine Behaviours

“slide 642”

Other Context/Machine Behaviours

“slide 643”

• • •

We refer to Appendix O (Pages 385–387) for an example related to the *inter-*
face requirements topic.

“slide 644”

“slide 645”

Dines Björner: 9th DRAFT: October 31, 2008

3.9.5 Machine Requirements

"SLIDE 646"

En Enumeration of Issues⁴⁰

"slide 648"

1 Performance Requirements	Page 136
(a) Storage Requirements	Page 136
(b) Machine Cycle Requirements	Page 136
(c) Other Resource Consumption Requirements	Page 136
2 Dependability Requirements	Page 136
(a) Accesability Requirements	Page 136
(b) Availability Requirements	Page 136
(c) Integrity Requirements	Page 136
(d) Reliability Requirements	Page 136
(e) Safety Requirements	Page 136
(f) Security Requirements	Page 136
3 Maintenance Requirements	Page 136
(a) Adaptive Maintenance Requirements	Page 136
(b) Corrective Maintenance Requirements	Page 136
(c) Perfective Maintenance Requirements	Page 136
(d) Preventive Maintenance Requirements	Page 136
4 Platform Requirements	Page 136
(a) Development Platform Requirements	Page 136
(b) Execution Platform Requirements	Page 136
(c) Maintenance Platform Requirements	Page 136
(d) Demonstration Platform Requirements	Page 136
5 Documentation Requirements	Page 136

"slide 649"

³⁹ "SLIDE 618"

⁴⁰ "SLIDE 647"

Performance Requirements⁴¹

*Machine Storage Consumption*⁴²

*Machine Time Consumption*⁴³

*Other Resource Consumption*⁴⁴

Dependability Requirements⁴⁵

*Accessability Requirements*⁴⁶

*Availability Requirements*⁴⁷

*Integrity Requirements*⁴⁸

*Reliability Requirements*⁴⁹

*Safety Requirements*⁵⁰

*Security Requirements*⁵¹

Maintenance Requirements⁵²

*Adaptive Maintenance Requirements*⁵³

*Corrective Maintenance Requirements*⁵⁴

*Perfective Maintenance Requirements*⁵⁵

*Preventive Maintenance Requirements*⁵⁶

Platform Requirements⁵⁷

*Development Platform Requirements*⁵⁸

*Execution Platform Requirements*⁵⁹

*Maintenance Platform Requirements*⁶⁰

*Demonstration Platform Requirements*⁶¹

Documentation Requirements⁶²

*Development Documentation*⁶³

Informative Documents

Prescription Documents

Analytic Documents

*Installation Documentation*⁶⁴

*Demonstration Documentation*⁶⁵

*User Documentation*⁶⁶

*Maintenance Documentation*⁶⁷

*Disposal Documentation*⁶⁸

⁴¹ "SLIDE 650"

⁴² "SLIDE 651"

⁴³ "SLIDE 652"

⁴⁴ "SLIDE 653"

⁴⁵ "SLIDE 654"

⁴⁶ "SLIDE 655"

⁴⁷ "SLIDE 656"

⁴⁸ "SLIDE 657"

⁴⁹ "SLIDE 658"

⁵⁰ "SLIDE 659"

⁵¹ "SLIDE 660"

⁵² "SLIDE 661"

⁵³ "SLIDE 662"

⁵⁴ "SLIDE 663"

⁵⁵ "SLIDE 664"

⁵⁶ "SLIDE 665"

⁵⁷ "SLIDE 666"

• • •

We refer to Appendix P (Pages 389–393) for an example related to the *machine requirements* topic.

“SLIDE 683”

“slide 681”

“slide 682”

3.10 Requirements Verification

“SLIDE 684”

TO BE WRITTEN

3.11 Requirements Validation

“SLIDE 685”

TO BE WRITTEN

3.12 Requirements Satisfiability and Feasibility

“SLIDE 686”

TO BE WRITTEN

3.13 Requirements Theory Formation

“SLIDE 687”

TO BE WRITTEN

3.14 Requirements Engineering Process Graph

“SLIDE 688”

TO BE WRITTEN

“slide 689”

3.15 Requirements Engineering Documents

“SLIDE 690”

TO BE WRITTEN

“slide 691”

3.15.1 Requirements Prescription Documents

TO BE WRITTEN

“slide 692”

64 “SLIDE 676”

65 “SLIDE 677”

66 “SLIDE 678”

67 “SLIDE 679”

68 “SLIDE 680”

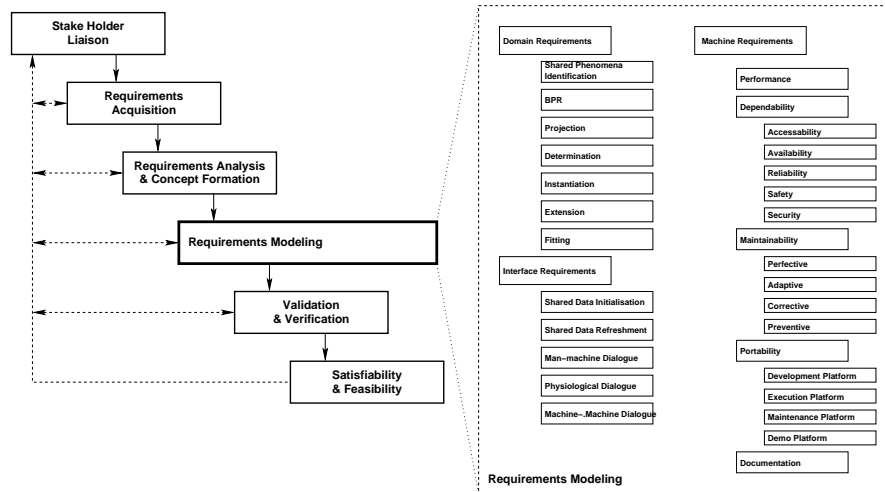


Fig. 3.1. Requirements engineering process graph

- 1 Stakeholders
- 2 The Acquisition Process
 - (a) Studies
 - (b) Interviews
 - (c) Questionnaires
 - (d) Indexed Description Units
- 3 Rough Sketches (Eurekas, IV)
- 4 Business Process Re-engineering
 - Sanctity of Intrinsic
 - Support Technology
 - Management and Organisation
 - Rules and Regulations
 - Human Behaviour
 - Scripting
- 5 Terminology
- 6 Facets: PG.:693
 - (a) Domain Requirements
 - Projection
 - Determination
 - Instantiation
 - Extension
 - Fitting
 - (b) Interface Requirements
 - Shared Phenomena and Concept Identification
 - (c) Mach. Req. PG.:694
 - Performance
 - ★ Storage
 - ★ Time
 - ★ Software Size
 - Dependability
 - ★ Accessibility
 - ★ Availability
 - ★ Reliability
 - ★ Robustness
 - ★ Safety
 - ★ Security
 - Maintenance
 - ★ Adaptive
 - ★ Corrective
 - ★ Perfective
 - ★ Preventive
 - Platform (P)
 - ★ Development P
 - ★ Demonstration P

- ★ Execution P
- ★ Maintenance P
- Doc. Reqs.
- Other Requirements
- (d) Full Requirements Facets Documentation

“slide 695”

3.15.2 Requirements Analysis Documents

TO BE WRITTEN

- | | |
|---|--|
| 2 Requirements Analysis and Concept Formation | (b) Model Checks |
| (a) Inconsistencies | (c) Test Cases and Tests |
| (b) Conflicts | 5 Requirements Theory |
| (c) Incompletenesses | 6 Satisfiability and Feasibility |
| (d) Resolutions | (a) Satisfaction: correctness, unambiguity, completeness, consistency, stability, verifiability, modifiability, traceability |
| 3 Requirements Validation | (b) Feasibility: technical, economic, BPR |
| (a) Stakeholder Walkthroughs | |
| (b) Resolutions | |
| 4 Requirements Verification | |
| (a) Theorem Proofs | |

3.16 Summary

“SLIDE 696”

TO BE WRITTEN

MORE TO COME

3.17 Exercises

Exercise 19. kap3.xs.1:

Solution 19 Vol. II, Page 436, suggests a way of answering this exercise.

Exercise 20. kap3.xs.2:

Solution 20 Vol. II, Page 436, suggests a way of answering this exercise.

Exercise 21. kap3.xs.3:

Solution 21 Vol. II, Page 436, suggests a way of answering this exercise.

Exercise 22. kap3.xs.4:

Solution 22 Vol. II, Page 436, suggests a way of answering this exercise.

Exercise 23. kap3.xs.5:

Solution 23 Vol. II, Page 437, suggests a way of answering this exercise.

Exercise 24. kap3.xs.6:

Solution 24 Vol. II, Page 437, suggests a way of answering this exercise.

Exercise 25. kap3.xs.7:

Solution 25 Vol. II, Page 437, suggests a way of answering this exercise.

Exercise 26. kap3.xs.8:

Solution 26 Vol. II, Page 437, suggests a way of answering this exercise.