Topic 1

The Triptych Paradigm

• The **prerequisite** for following this part of the lecture is that you have at least some introductory level programming skills, as, for example, obtained through a year of Java or C# programming.

TWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering
s Triptych Paradigm			Institute of Informatics and Mathematical Modelling Technical University of Denmark
se/db/vollI/3ch1/3ch1-i	April 5, 2006, 09:00	Page 3, Topic: 1, Foil: 3	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark

- The **objective** is to make the reader a professional software engineer with respect to understanding the crucial phases, stages and steps of software development.
- The **treatment** is precise but informal.

WARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering
1 Friedrich L. Bauer 1968			Institute of Informatics and Mathematical Modelling Technical University of Denmark
e/db/vollI/3ch1/3ch1-i	April 5, 2006, 09:00	Page 5, Topic: 1, Foil: 5	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark

Friedrich L. Bauer 1968

Software engineering is the establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines.

So we are left to find out what is meant by engineering principles. These "engineering principles" cannot just be those of conventional engineering as we think that the engineering of software is radically different from other engineerings. Conventional engineering builds on the laws of physics. Software engineering builds on mathematics, notably algebra and logic.

TWARE ENGINEERING: Domains, Requirements and Sc	oftware Design	Volume 3	Department of Computer Science and Engineering	DTU
2 Ian Sommerville 1980-2000			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
ne/db/vollI/3ch1/3ch1-i	April 5, 2006, 09:00	Page 7, Topic: 1, Foil: 7	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	

- Engineering discipline:
 - \star Engineers make things work.
 - ★ They apply theories, methods and tools where these are appropriate but they use them selectively and always try to discover solutions to problems even when there are no applicable theories and methods to support them.
 - ★ Engineers also recognise that they must work to organisational and financial constraints so they look for solutions within these constraints.

- The aims are to introduce the basic ideas of
 - * domain engineering,
 - \star requirements engineering and
 - ⋆ software design

as they relate to one another, to introduce the concept of separation of concerns as represented here by the concepts of

- \star stages and
- $\star\,steps$

of development, and thus to introduce the concept of the triptychsoftware development process model.

.1.1 "Old" Delineations	nu and Salman Delign	Totalin 3	Department of Computer Science and Engineering Institute of Informatics and Mathematical Modelling Technical University of Desmark	#
home/db/vol11/3ch1/3ch1-i	April 5, 2006, 09:00	Page 4, Topic: 1, Foil: 4	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	- 1
Delineations of Software Engineering				
T	//OL 19. D. 1	. ,.		

"Old" Delineations • The term "software engineering" seems to have many meanings.

• We shall bring in some of the characterisations that are given in previous textbooks as well as from elsewhere.

OFTWARE ENGINEERING: Domains, Requirements and	Volume 3	1	
.1.1.2 Ian Sommerville 1980-2000			1
home/db/vallI/3ch1/3ch1-i	April 5, 2006, 09:00	Page 6, Topic: 1, Foil: 6	1

Ian Sommerville 1980-2000

Software engineering is an engineering discipline

- which is concerned with all aspects of software production
- from the early stages of system specification
- through to maintaining the system after it has gone into use.

In this definition, there are two key phrases:

• All aspects of software production:

- \star Software engineering is not just concerned with
- \diamond the technical processes of software development
- but also with activities such as software project management
- and with the development of tools, methods and theories to support software production.

We are getting some engineering principles unveiled, albeit of the conventional kind.

IEEE Std. 610.12-1990

The IEEE's Standard Glossary of Software Engineering Terminology:

Software engineering is defined as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.

Again, a very conventional engineering characterisation.

© Dines Rigemer, Fredoxej 11, DH-2040 Holte, Denma

E-maile: defilimm.dtu.dx, bjorner@gmail.com, dinee@bjorner.biz; URL: www.imm.dtu.dk/"

TWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DTU
.5 Shari Lawrence Pfleeger, 2001			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
нe/db/vollI/3ch1/3ch1-i	April 5, 2006, 09:00	Page 11, Topic: 1, Foil: 11	Richard Patersens Plads, DK-2000 Kgs Lyngby, Denmark	
	Choni I omnono	e Pfleeger, 2001		1

As software engineers, we use our knowledge of computers and computing to help solve problems . . . identification of problems and of when a computing solution may be appropriate, further analysis of such problems, and synthesis of solutions using method principles, techniques and tools, are ingredients of software engineering.

We are not getting much closer to our claimed difference between conventional engineerings and software engineering. Pfleeger's characterisation is OK, but insufficient.

525 3720, Fax: +45 4593 0

© Dines Bijørner, Fredovej 11, DH-2840 Holte, Denre

E-mails: doffenn.dtu.ds, hjorner@gmail.com, diest@hjorner.biz; URL: www.imm.dtu.di

TWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DTU
7 Accreditation Board for Engineering and Technology (ABET)		Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
ne/dlb/voll1/3ch1/3ch1-i	April 5, 2006, 09:00	Page 13, Topic: 1, Foil: 13	Richard Petersens Plads, DK-2000 Kgs Lyngby, Denmark	_
<u> </u>				- /

Accreditation Board for Engineering and Technology (ABET)

Engineering is the profession in which a knowledge of the mathematical and natural sciences gained by study, experience and practice is applied with judgment to develop ways to utilize, economically, the materials and forces of nature for the benefit of mankind

5 3720, Fax: +45 4593 0074

© Dines Bijørner, Fredoug 11, DK-2940 Holte, Denma

E-mails: dellimmetra dis, bjorner@gmail.com, dimer@bjorner.biz; URL: www.imm.dtu.dk/*o

TWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DTU
Our View: What Is Software Engineering?			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
re/dlb/vollI/3ch1/3ch1-i	April 5, 2006, 09:00	Page 15, Topic: 1, Foil: 15	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denesark	

Our View: What Is Software Engineering?

- We shall characterise the concept of 'software engineering' as follows:
 - \star Software engineering is the establishment and use of sound methods for the efficient construction of efficient, correct, timely and pleasing software that solves the problems such as users identify them.
 - \star Software engineering extends the field of computing science to include also the concerns of building of software systems that are so large or so complex that they necessarily are built by a team or teams of engineers.

| Digital Continues | Digi

David Lorge Parnas

Software engineering is defined as the multi-person construction of multi-version software.

This is, of course, not all that Parnas has to say about software engineering. As much of his other musings this one is cogent.

15 4525 3720, Fax: +45 4593 0074

S) Dines Rjørner, Fredovej 11, DK-2840 F

E-mail: debinen.da.dk, bjorner@gmail.com, diner@bjorner.biz; URL: www.imm.da.dk/"di

| PFTWARE ENGINEERING: Domains, Requirements and Software Design | Volume 3 | Opposite of Compute Science of Engineering Line of Engineering Line of Computer Science of Engineering Line of Science of Engineering Line of Engine

Carlo Ghezzi, Mehdi Jazayeri and Dino Mandrioli, 2002

Software engineering is the field of computer science that deals with the building of software systems that are so large or so complex that they are built by a team or teams of engineers.

This definition hides the real content in its reference to computer (including computing) science, i.e., the mathematical discipline upon which the software engineers work. But, as for Parnas' characterisation, the Ghezzi/Jazayeri/Mandrioli characterisation emphasises scale.

+45 4525 3720, Fax: +45 4593 0074

© Dines Rjørner, Fredoxej 11, DK-2940 Halte, Dec

E-mails: deliferen.dts.dk, bjorner@graal.com, direr@bjorner.biz; URL: www.imm.dts.dk/

 OF TWINE ENGINEED/IK. Domain, Requirements and Software Disages
 Volume 3

 1.18 Hars van Viet
 April 5, 2006, 09.00
 Page 14, Topic 1, Foli 14

Hans van Vliet

- \bullet Software engineering concerns the construction of large programs.
- The central theme is mastering complexity.
- Software evolves.
- The efficiency with which software is developed is crucial.
- Regular co-operation between people is an integral part of programming-in-the-large.
- The software has to support its users effectively.
- Software engineering is a field in which members of one culture create artifacts on behalf of members of another culture.

: +45 4525 3720, Fax: +45 4593 0074

© Diess Rjømer, Fredovij 11, DK-2840 Hulte, Dec

OFTWARE ENGINEERING, Domains, Requirements and Software Dasign Volume 3

1.2 Our View: What is Software Engineering?

April 5, 2006, 06:00 Page 16, Topic: 1, Folt: 16

Department of Computer Science and Engineering Institute of Informatics and Mathematical Modelling Technical University of Denmark Richard Protesson Plads, DK-2000 Kgs Lyngby, Den

- ★ Software engineering is the profession in which a knowledge of mathematics, gained by study, experience and practice, is applied, with judgment to develop ways of exploiting mathematics to (i) understand the problem domain, (ii) the problem and (iii) to develop computing systems, especially software solutions, to such problems as are conveniently solved by computing.
- * Software engineering thus consists of (i) domain engineering (in order to understand the problem domain), (ii) requirements engineering (in order to understand the problem and possible frameworks for their solution) and (iii) software design (in order to actually implement desired solutions).

E-maile: diellimmetrude, bjorner@grazil.com, diese@bjorner.biz; UFE: www.imm.dtu.dk/"db

O Providence Code of the Part Mark Hale Providence

E-mails: delinen das dis hjomerbynal com, denobljomer hiz URL: www.imm.dts.dh/*d

In the next part of the lecture we shall examine these three concepts:

* domain engineering,

* requirements engineering and

* software design.

.....

Dines Bijamer, Fredoug 11, DW-2040 Holte, Denn.

E-maile: dollimm.dtu.dk; bjorner@gmail.com, dinee@bjorner.blu; URL: www.imm.dtu.dk/

TWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Degartment of Computer Science and Engineering Institute of Informatics and Mathematical Modelling Technical University of Desmank	DTU
.1 Theme of this Lecture				DTU
не/dlb/vollI/3ch1/3ch1-ё-і	April 5, 2006, 09:00	Page 19, Topic: 2, Foil: 2	Richard Petersens Plads, DK-2000 Kgc Lyngby, Denmark	[]
0 0 1 1		1: 4: 1 :		

- \bullet So, from descriptions of the application domain
 - \star we construct prescriptions of the requirements;
- and from prescriptions of the requirements
 - ⋆ we design the software.

25 3720, Fax: +45 4593 007

(2) Dines Bijørner, Fredoxej 11, DK-2840 Holte, Denmar

E-mails: dbffirm.dtu.dk, bjorner@gmail.com, dines@bjorner.biz; URL: www.imm.dtu.dk)

TWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DTU
On Universes of Discourse and Domains			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
ne/dlb/vollI/3ch1/3ch1-ii-i	April 5, 2006, 09:00	Page 21, Topic: 2, Foil: 4	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Dwanark	/

On Universes of Discourse and Domains

- Above we have used the term "application domain" without explaining what we mean by that term.
- We shall now explain that term as well as the more general term 'universe of discourse' and the simpler term 'domain'.

Characterisation 1.1 By a universe of discourse we shall understand anything that can be spoken about.

3720, Fax: +45 4593 0074

© Dines Bigener, Fredoxy 11, DH-2840 Holte, Denns

E-mails: dollimmétu.ds, bjorner@gmail.com, dinnel@bjorner.biz; UPE: www.imm.dtu.dk/"d

PWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DT
On Universes of Discourse and Domains			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
ie/db/volll/3ch1/3ch1-ii-i	April 5, 2006, 09:00	Page 23, Topic: 2, Foil: 6	Richard Petersent Plads, DK-2800 Kgc.Lyngby, Denmark	
. TDL . : 1 . C :	-1 £ (41::)			

- The indefinite class of "things" to which computing may be applied, i.e., application domains (see next).
- The infinite class of anything else that does not satisfy the above characterisations. Examples are: philosophy, politics, poetry, etc.

Characterisation 1.2 By an application domain we shall understand anything to which computing may be applied.

DePTWARE ER/OKEE/DIKC Domains, Requirements and Software Dosign

20.1 Themse of this Lecture

2.0.1 Themse of this Lecture

April 5, 2006, 00:00 Page 18, Topic 2, Falt 1

Topic 2

Topic 2

Topic 2 The Triptych of Software Engineering

Theme of this Lecture

- Before some specific software can be designed and coded,
- we must understand the requirements
- that this software must fulfill.
- Before requirements can be written down,
- \bullet we must understand the application domain
- for which the software is to be developed.

TWARE ENCINEERING. Commiss, Requirements and Software Design

1. There of this Lessore

1. There of this Lessore

- \bullet Ideally speaking we would wish to proceed from
 - \star describing the application domain,
 - \star via prescribing the requirements,
 - \star to implementing the software.
- Actual life sometimes forces us,
 - \star and always permits us,
 - \star to iterate between these three phases of software development.

2.1 On Universe of Discourse and Domains

April 5, 2006, 09:00 Page 22, Topic 2, Foil: 5

Example 1.1 *Universes of Discourse*: We shall take the view here that there are basically three classes of universes of discourse:

- The definite, but singleton class of software engineering as an intellectual concept, i.e., software development in general and programming in particular.
 - \star So, domain engineering, requirements engineering and software design could each, or as a whole, be a universe of discourse.
 - \star These lectures take this intellectual concept of software engineering as its universe of discourse.
 - \star As an intellectual concept the software engineering universe of discourse is not an application domain.

3220, Fax: +45 4593 0074 © Direc Rjemer, Fredovrj 11, DK-2860 Halte, 0

E-mails: debleme das dis, bjørnerbynal com, die erbbjørner bla; URL: www.imm.dts.ds/

 OFTWARE ENGINEERING. Domains, Requirements and Software Design
 Volume 3

 2.1 On Universe of Discourse and Domains
 April 5, 2006, 09.00
 Page 24, Topic 2, Folk 7

 home(dh/vell1/bb1/bb1/bb1)
 April 5, 2006, 09.00
 Page 24, Topic 2, Folk 7

Technical University of Denmark Richard Protestens Plads, DK-2800 Kgs.Lyngby, Denmark

Example 1.2 Application Domains: We shall take the view that there are basically three classes of application domains:

 The class of applications which can be characterised as supporting the teaching or study of a subject field: educational or training software, respectively experimental software for theorem proving, or the like. TWARE ENGINEERING. Domains, Requirements and Software Design

Volume 3

On Universes of Discourse and Domains

 The class of applications which can be characterised as supporting the development of computing systems themselves: compilers, operating systems, database management systems, data communication systems, etc.

And the class of applications which can be characterised as not supporting the development of computing systems themselves, but that of business, or industry software.

Characterisation 1.3 By a *domain* we mean an application domain.

That is, the two terms "application domain" and "domain" are taken to be synonymous.

© Dines Bjørner, Fredovej 11, DH-2940 Holte, Denmi

E-mails: dirBinon.dtu.dk, bjorner@graail.com, dines@bjorner.biz; URL: www.inen.dtu.dk/"d

On Universes of Discourse and Domains		ı	Institute of Informatics and Mathematical Modelling Technical University of Denmark	#
ne/db/vollI/3ch1/3ch1-ii-i	April 5, 2006, 09:00	Page 27, Topic: 2, Foil: 10	Richard Petersens Plads, DK-2800 Kgs.Lyngby, Denmark	_ /
. Th li i		l.: 4l C :	.1	

 The applications of software within the financial services sector: banks, insurance companies, securities trading (stock and bond exchanges, traders, brokers), portfolio and investment management, venture capital companies, etc., individually define application domains, and together "define" the financial service industry as a domain.

525 3720, Fac: +45 4593 0

© Dines Rijamer, Fredoxej 11, DK-2840 Holte, Denmari

E-mails: dollinun.dtu.ds, bjorner@gmail.com, diese@bjorner.biz; UFL: www.imm.dtu.d

THAT ENGINEERING DOMAIN, requirements and Johnson Design		Volum 3	Department of Computer Science and Engineering	DIO
On Universes of Discourse and Domains			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
re/db/voll1/3ch1/3ch1-ii-i	April 5, 2006, 09:00	Page 29, Topic: 2, Foil: 12	Richard Patentens Plads, DK-2000 Kgs.Lyngby, Denmark	
				-1

• The applications of software within the machining (metal-working) manufacturing sector individually define application domains, and together "define" machining (metal-working) manufacturing as a domain.

IS 3720, Fax: +45 4593 0074

② Dines Bjørner, Fredoxý 11, DK-2940 Holte, Denma

E-mails: definends uds, bjorner@gmail.com, direc@bjorner.biz; UFE: www.inm.dtu.dk/"db

WARE ENGINEERING: Domains, Requirements and Software Design I. General		Volume 3	Department of Computer Science and Engineering Institute of Informatics and Mathematical Modelling Technical University of Denmark	DTU
Characterisatio	on 1.5 (II) By de	Page 31, Topic: 3, Foil: 2	Richard Pronunce Plade, DK-2000 Kgc Lynghy, Denmark	-/

Characterisation 1.5 (II) By domain description we shall also mean the process of domain capture, analysis and synthesis, and the document which results from that process.

Characterisation 1.6 By domain engineering we mean the engineering of domain descriptions, that is, of their development: (i) from domain capture and analysis (ii) via synthesis, i.e., the domain description document itself, (iii) to its validation with stakeholders and its possible theory development.

Department of Computer Science and Engineering Institute of Informatics and Mathematical Modelling Technical University of Denmark Sichard Patersens Plads, DK-2000 Kgs.Lyngby, Denmark

Example 1.3 *Domains:* We continue our exemplification of (application) domains — of the third class mentioned just above.

 The applications of software within the transportation sector: Railways, airlines, shipping, public and private road transport (buses, taxis, trucks, automobiles in general), etc., individually define application domains, and together "define" transportation as a domain.

4525 3720, Fax: +45 4593 00

es Rjørner, Fredovej 11, DK-2940 Holte, Denn

mails: dolferen das de, bjornerfigmail.com, directibjorner biz; URL: www.imm.das.de/"do

¹Here our use of "define" indicates that we are not formally defining the subject term. We are merely giving a rough characterisation.

OFTWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DTU
.2.1 On Universes of Discourse and Domains			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
home/db/volII/3ch1/3ch1-ii-i	April 5, 2006, 09:00	Page 28, Topic: 2, Foil: 11	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	
				- /

• The applications of software within the healthcare sector: hospitals, family doctors (i.e., private, practicing physician), pharmacies, community nurses, retraining and convalescent clinics, the public health authorities, etc., individually define application domains, and together "define" healthcare as a domain.

525 3720, Fax: +45 4593 0074

Dines Rijamer, Fredovij 11, DK-2940 Hulte, Denr

 $E-mail: deblements.dx, bjorner bgmail.com, die sebbjorner biz; URL: www.imm.dxs.dx/^*dx$



 We shall give a brief characterisation of what we mean by domain engineering.

Characterisation 1.4 (I) By a domain description we shall understand a description of a domain, that is, something which describes observable phenomena of the domain: entities, functions over these, events and behaviours.

45 4525 3720, Fax: +45 4593

© Dines Rjørner, Fredoxej 11, DK-2840 Holte, Dec

E-maile: diebiene das die hjomerbymail.com, diesebhjomer hiz; URL: wew.imm.dtu.dk/

 OFTWARE ENGINEERING. Domains, Requirements and Software Design
 Volume 3

 2.2.1 General
 April 5, 2006, 00:00
 Page 32, Topic 3, Folt 3

Technical University of Denmark Richard Petersens Plads, DK-2900 Kgs.Lyngby, Den

- \bullet So what does it mean to understand the application domain?
- \bullet To us it means that we have described it.
- \bullet That the description is consistent, i.e.,
 - \star does not give rise to contradictions,
- \bullet and that the description is relatively complete, i.e.,
 - \star does describe "all the things" needed to be described.

4525 3720, Far: +45 4593 0074

© Dines Bijørner, Fredovej 11, DK-2940 Hobe, Denmark

E-maile: dirBinon.deu.de, bjorner@granl.com, direc@bjorner.biz; URL: www.inem.deu.de/"db

+45 4525 3720, Fax: +45 4593 0074

O Proc Proces Code of 11 Dec 20th links December

E-mails: dieBenn das dis, bjonnet@mail.com, die nothjonne biz; URL: www.imm.dta.ds/"db

What Do We Expect from a Domain Description?			Institute of Informatics and Mathematical Modelling Technical University of Denmark	≕
db/vallI/3ch1/3ch1-ii-d	April 5, 2006, 09:00	Page 33, Topic: 3, Foil: 4	Richard Petersens Plads, DK-2000 Kgs Lyngby, Denmark	
Wh	at Do We Expect fro	om a Domain Desc	cription?	/
• What must w	e expect from a d	omain descripti	on?	
	с схрест пош а с	omam descripti	on:	
• We expect				
⋆ that it desc	ribes the applicat	ion area as it is	S.	
IS 3720, Fac: +45 4593 0074	© Dines Rigerer, Fredoug 11, DW-2840 Holte, Desc	nak E	mails: dollinen.dtu.dk, bjorner@grazi.com, dines@bjorner.biz; UPL: www	ow imm dtu dk)
ARE ENGINEERING: Domains, Requirements and	d Software Design	Volume 3	Department of Computer Science and Engineering Institute of Informatics and Mathematical Modelling	DTU
What a Domain Description Does Describe db/voll1/3ch1/3ch1-ii-d	April 5, 2006, 09:00	Page 35, Topic: 3, Foil: 6	Institute of Informatics and Mathematical Modelling Technical University of Denmark Richard Petersen Plads, DK-2800 Kgs Lyngby, Denmark	Ħ
F				
Ĺ	What a Domain Des	cription Does Des	cribe	/
• So what does	a domain descript	tion contain?		
	•			
• To us a doma	in description con	tains:		
* descriptions	s of the phenome.	na that can be	observed, that can	
be physical	ly sensed, in the d	lomain, and		
* description	s of the concepts,	i.e., the abstra	ctions that these	
-	"embody".	. ,		
F				
5 3720, Fac: +45 4593 0074	© Dines Rigerer, Fredox 11, DK-2840 Holte, Desc	nak E	male delimentude, hjorerligmal.com, dinselhjorer bit UFL wa	ow imm dtu dk
ARE ENGINEERING: Domains, Requirements and	d Software Design	Volume 3	Department of Computer Science and Engineering	DTU
ARE ENGINEERING: Domains, Requirements and Pormain Phenomena and Concepts	I Software Design April 5: 2006, 09:00	Volume 3 Page 37, Topic: 3, Foil: 8	Institute of Informatics and Mathematical Modelling Technical University of Denmark	DII
	J Software Design April 5, 2006, 09:00		Institute of Informatics and Mathematical Modelling	DIL
		Volume 3 Page 37, Topic: 3, Fol: 8	Institute of Informatics and Mathematical Modelling Technical University of Denmark	DITE
lb/vollI/3ch1/3ch1-≅-d	• E)	ntities	Institute of Informatics and Mathematical Modelling Technical University of Denmark	<u>DII</u>
• Entities are "f	things" that one c	an point to,	Institute of Informatics and Machinerical Maching Technical Unionized Chemises Richard Processor Flork, DN-2020 Kgs Lyugh, Deneual Richard Processor Flork, DN-2020 Kgs Lyugh, Deneual	DTI
• Entities are "f	• E)	an point to,	Institute of Informatics and Machinerical Maching Technical Unionized Chemises Richard Processor Flork, DN-2020 Kgs Lyugh, Deneual Richard Processor Flork, DN-2020 Kgs Lyugh, Deneual	DTR
• Entities are "it • things that ty	things" that one copically become da	an point to,	Institute of Informatics and Machinerical Maching Technical Unionized Chemises Richard Processor Flork, DN-2020 Kgs Lyugh, Deneual Richard Processor Flork, DN-2020 Kgs Lyugh, Deneual	DTL
• Entities are "to things that ty • things that are	things" that one opically become date to have a type	an point to,	Institute of Informatics and Machinerical Maching Technical Unionized Chemises Richard Processor Flork, DN-2020 Kgs Lyugh, Deneual Richard Processor Flork, DN-2020 Kgs Lyugh, Deneual	DTT.
• Entities are "it • things that ty	things" that one opically become date to have a type	an point to,	Institute of Informatics and Machinerical Maching Technical University Olivania Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous	<u>DIT</u>
• Entities are "(• things that ty • things that ar	things" that one opically become date to have a type	an point to,	Institute of Informatics and Machinerical Maching Technical University Olivania Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous	
• Entities are "to things that ty • things that are	things" that one opically become date to have a type	an point to,	Institute of Informatics and Machinerical Maching Technical University Olivania Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous	<u>on</u>
• Entities are "6 • things that ty • things that ar	things" that one opically become date to have a type	an point to,	Institute of Informatics and Machinerical Maching Technical University Olivania Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous	<u>DTM</u>
• Entities are "to things that ty • things that are	things" that one opically become date to have a type	an point to,	Institute of Informatics and Machinerical Maching Technical University Olivania Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous	DIM
• Entities are "6 • things that ty • things that ar	things" that one opically become date to have a type	an point to,	Institute of Informatics and Machinerical Maching Technical University Olivania Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous	DTM
• Entities are "to things that ty • things that are	things" that one opically become date to have a type	an point to,	Institute of Informatics and Machinerical Maching Technical University Olivania Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous	DTR
• Entities are "to things that ty • things that are	things" that one opically become date to have a type	an point to,	Institute of Informatics and Machinerical Maching Technical University Olivania Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous	DIN
• Entities are "6 • things that ty • things that ar	things" that one opically become date to have a type	an point to,	Institute of Informatics and Machinerical Maching Technical University Olivania Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous	DIN ##
Entities are "to things that ty things that are and a value (continuous).	things" that one opically become date to have a type	an point to,	Institute of Informatics and Machinerical Maching Technical University Olivania Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous	
Entities are "to things that ty things that are and a value (continuous).	things" that one copically become date to have a type of that type).	an point to,	Institute of Informatics and Machinerical Maching Technical University Olivania Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous	DIR III
Entities are "things that ty things that are and a value (continuous).	things" that one copically become date to have a type of that type).	an point to,	Institute of Informatics and Machinerical Maching Technical University Olivania Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous Richard Processor Flork, Div. 2020 Kgs Lyugh, Denous	DIT III
Entities are "things that ty things that are and a value (continuous).	things" that one copically become date to have a type of that type).	an point to,	makes the fundament and telescope te	no insendina dia
Entities are "(things that ty things that ar and a value (things that ar	things" that one copically become date to have a type of that type).	an point to, ta "inside" a co	make a financian of Managara M	ma insens disa di
Entities are "(things that ty things that ar and a value (things that ar	things" that one copically become date to have a type of that type).	an point to,	mainter de formación and Materianica de Materianica	ma insens disa di
Entities are " things that ty things that ar and a value (The retion of the second of the s	things" that one copically become date to have a type of that type).	an point to, ita "inside" a co	make a financian of Managara M	ma insens disa di
Entities are "for things that ty things that are and a value (state of the second	things" that one copically become date to have a type of that type).	an point to, ita "inside" a co	make a financian of Managara M	ma insens disa di
Entities are " things that ty things that ar things that ar and a value (SER, for the ENGINEERING Durains, Regiments and Durain (Mark (MARK) (MA	things" that one copically become day to have a type of that type).	an point to, ita "inside" a co	make of temporal between the temporal baseling between the form of the temporal baseling between the temporal baseling baseling between the temporal baseling baseling between the temporal baseling baselin	ma insens disa di
Entities are " things that ty things that ar things that ar and a value (STOR, For. +44 COLD ENT. ARE ENCORPERING. Domains. Requirements are bounded (Parkell Stoke). We shall later type Harbour, Ship,	things" that one copically become date to have a type of that type).	an point to, ita "inside" a co	make of temporal between the temporal baseling between the form of the temporal baseling between the temporal baseling baseling between the temporal baseling baseling between the temporal baseling baselin	ma insens disa di
Entities are " things that ty things that ar and a value (ARE ENGINEERING Domain, Requirements, and Domain Phatamana and Concepts Revoil (Phatamana and Concepts Revoil	things" that one copically become date to have a type of that type).	an point to, ita "inside" a co	make of temporal between the temporal baseling between the form of the temporal baseling between the temporal baseling baseling between the temporal baseling baseling between the temporal baseling baselin	ma insens disa di
Entities are " things that ty things that ar and a value (ARE ENGINEERING. Domains, Requirements are Showing Photometers of Concepts Report Report of Concepts Report Report of Concepts Report Report of Concepts Report Repo	things" that one copically become date to have a type of that type). © One Spiere, Frankry II, DO 200 NANO, One April 1, 2000, 60:00 explain the notate HoldArea, Buoy,	an point to, ita "inside" a co	make of temporal between the temporal baseling between the form of the temporal baseling between the temporal baseling baseling between the temporal baseling baseling between the temporal baseling baselin	ma insens disa di
Entities are " things that ty things that ar and a value (The HALLES H	things" that one copically become date to have a type of that type). © One Spiers, Francy 11, 06:388 Nate, One April 5, 2006, 09:00 explain the notate HoldArea, Buoy, bour → Ship-set Harbour → Hold	an point to, ita "inside" a co	make of temporal between the temporal baseling between the form of the temporal baseling between the temporal baseling baseling between the temporal baseling baseling between the temporal baseling baselin	ma insens disa di
Entities are " things that ty things that ar and a value (AME ENGINEERING Domains, Requirements are Showill Plantomers and Consepts Reveall Plantomers and Consepts Reveal Plantomers Reveal Plantomers Reveal Plantomers Reveal Plantomers Reveal Plantomers Reveal Plantomers Reveal P	things" that one copically become date to have a type of that type). © One Spiere, Frankry II, DO 200 NANO, One April 1, 2000, 60:00 explain the notate HoldArea, Buoy,	an point to, ita "inside" a co	make of temporal between the temporal baseling between the form of the temporal baseling between the temporal baseling baseling between the temporal baseling baseling between the temporal baseling baselin	/ /
Entities are "1 things that ty things that ar things that ar and a value (We shall later type Harbour, Ship, value obs_Ships: Har obs_Buoys: Har obs_Buoys: Har	things" that one copically become date to have a type of that type). © One Spiers, Francy 11, 06:388 Nate, One April 5, 2006, 09:00 explain the notate HoldArea, Buoy, bour → Ship-set Harbour → Hold	an point to, ita "inside" a co	make of temporal between the temporal baseling between the form of the temporal baseling between the temporal baseling baseling between the temporal baseling baseling between the temporal baseling baselin	/ /
Entities are "1 things that ty things that ar things that ar and a value (We shall later type Harbour, Ship, value obs_Ships: Har obs_Buoys: Har obs_Quays: Har obs_Quays: Har	things" that one copically become date to have a type of that type). **Software Design** **April 2, 2006, 59:00 **Explain the notate thought and the copies of the copi	an point to, ita "inside" a co la "inside" a co	make of temporal between the temporal baseling between the form of the temporal baseling between the temporal baseling baseling between the temporal baseling baseling between the temporal baseling baselin	ma insens disa di
Entities are "for things that ty things that are and a value (or things that are the same that are	things" that one copically become date to have a type of that type). Some Spine Finding II. DE 288 NATE, Compared to the comp	an point to, ita "inside" a co	make of temporal between the temporal baseling between the form of the temporal baseling between the temporal baseling baseling between the temporal baseling baseling between the temporal baseling baselin	/ /
Entities are "for things that ty things that are and a value (of things that are and things that are the same and the same are the same and the same are the same and the same are the same	things" that one copically become date to have a type of that type). Sometime Daries April 2, 2006, 59 000 explain the notate HoldArea, Buoy, bour Buoy-set Harbour Hold-bour Buoy-set Harbour Quay-set CSA-set	an point to, ta "inside" a co volume 3 Propo 30, Trapic 3, Trail 10 cition now used: Quay, CSA, Po lArea-set tet	make of temporal between the temporal baseling between the form of the temporal baseling between the temporal baseling baseling between the temporal baseling baseling between the temporal baseling baselin	ma insens disa di
Entities are "for things that ty things that are and a value (of things that are and things that are and things that are the same and the same are the same and the same are th	things" that one copically become date to have a type of that type). Sometime Daige April 5, 2006, 2000 explain the notate HoldArea, Buoy, bour — Ship-set Harbour — Hold chour — GSA-set hip — Position-set	an point to, ta "inside" a co volume 3 Propo 30. Trajec 3. Trajec 30 cition now used: Quay, CSA, Po lArea-set tet et eet	make of temporal between the temporal baseling between the form of the temporal baseling between the temporal baseling baseling between the temporal baseling baseling between the temporal baseling baselin	Dies der de de

obs_Position: HoldArea \rightarrow Position-set

What a Domain Description Does Not Describe

- \bullet To us "as it is" means
 - \star that we have described it without any reference to requirements to any new computing system (i.e., software),
 - \star let alone to any (implementation, etc.) of such a new computing system (i.e., software).
- \bullet The above was expressed in terms of what a domain description does not contain.

Domain Phenomena and Concepts

- \bullet What are the phenomena and concepts alluded to just above?
- \bullet To us these phenomena and concepts are such as:
 - * entities,
 - \star functions,
 - \star events and
 - * behaviours.
- We overview these four categories of phenomena and concepts.

Example 1.4 Entities: For a domain of harbours some typical entities are:

- \bullet ships,
- holding area(s) where ships may wait for a buoy or a quay position,
- buoys,
- \bullet quay positions and
- cargo storage areas.
- The harbour can be considered an entity composed from the above.

- From a harbour one can observe all the
 - * ships in the harbour,
 - \star holding areas of the harbour,
 - \star buoys of the harbour,
 - \star quay positions of the harbour and all the \star container storage areas of the harbour.
- \bullet Positions are associated with
- $\star \text{ ships,}$
- \star holding areas,
- \star buoys and
- * quays.

We sketch and explain the following formal text: type $A,\,B,\,C,\,...,\,P,\,Q,\,R,\,...$ value a:A, obs_B: $A \rightarrow B$ obs_C: B \rightarrow C obs_Ps: B \rightarrow P-set obs_Ql: $C \to Q^*$

FWARE ENGINEERING: Domains, Requirements and	Software Design	Volume 3	Department of Computer Science and Engineering	
.5 Domain Phenomena and Concepts			Institute of Informatics and Mathematical Modelling Technical University of Denmark	
н/db/vollI/3ch1/3ch1-ii-d	April 5, 2006, 09:00	Page 43, Topic: 3, Foil: 14	Richard Petersens Plads, DN-2000 Kgs.Lyngby, Denmark	
			/	

Example 1.5 Functions: For a domain of harbours some typical functions are:

(i) An arriving ship asks the harbour whether it can be allocated either a holding area, a buoy or a quay position.

 $value: inquire: Ship \times Harbour \rightarrow Bool$

(ii).1 An arriving ship which can be allocated a holding area, a buoy, or a quay position requests the position.

value: request: Ship×Harbour → Position

I WARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DIO
.5 Domain Phenomena and Concepts			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
и/db/vollI/3ch1/3ch1-ii-d	April 5, 2006, 09:00	Page 45, Topic: 3, Foil: 16	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	
				-/

We define three sorts, i.e., abstract types, and give the signature of four functions:

$_{\mathrm{type}}$

(0) A, B, C

value

- (1) inv_A: $A \rightarrow \mathbf{Bool}$
- (2) obs_B: $A \rightarrow B$
- (3) gen_C: $B \rightarrow C$
- (4) chg_B: $A \times B \rightarrow B$

WARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	
5 Domain Phenomena and Concepts			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
e/db/vollI/3ch1/3ch1-ii-d	April 5, 2006, 09:00	Page 47, Topic: 3, Foil: 18	Richard Petersens Plads, DK-2000 Kgs Lyngby, Denmark	/
/.v . =				- /

- (4) chg_B is intended as an operation (i.e., a generator function): from Cartesian values over A and B it generates values of type B intended to replace the argument of type B. One might thus write any of the below:
 - **variable** b:B := ...; ... b:= $chg_B(a,b)$...
 - let $b'=chg_B(a,b)$ in ...; b:=b'; ... end
 - let $b'=chg_B(a,b)$ in ... $gen_C(b')$... end

• Functions

- Phenomena or concepts could be
- \bullet functions that apply to entities and
 - * either test for some property,
 - * observe some subentity, i.e., yield a data value that is "computed" from such entities, or
 - \star actually change the entity value in which case we call the function an operation, or an action.

(ii).2 For a ship destined for a quay position one needs to know how

many containers to unload to and how many to load from the harbour: $value: unload_load_quantities: Ship \times Harbour \rightarrow Nat \times Nat$

(iii) A ship [un]loading some cargo.

value: [un]load: Ship×Quay → Ship×Quay

ullet (0) A, B and C are sorts, i.e., further unspecified abstract types.

- (1) inv_A is a predicate: it is supposed to yield **true** for well-formed values of A, false otherwise.
- \bullet (2) obs_B is intended as an observer function: from values a of sort A it observes, i.e., extracts, values of type B that are somehow "contained" in a.
- (3) gen_C is intended as a generator function: from values of sort B it computes values of type C.

Example 1.6 The A, B, Cs of Ships and Harbours: The reader is asked to complete this example, that is, to relate the types and functions of Example 1.5 to the sorts and functions of the box above!

• Events

- Events happen, i.e., occur. And when events occur they do so instantaneously.
- \bullet Events may convey information, i.e., have significance other than just occurring.
- We can speak of external events and of internal events.

Example 1.7 Events: For a domain of harbours some typical events

- a ship arrives at a harbour;
- a ship declares itself ready to unload or to load;
- a ship and a quay engage in the events of unloading and loading;
- a ship declares itself ready to depart a holding area, or a buoy or a quay position.

channel sh,hs:MSG sqr:ArrDep sq,qs:Cargo value $ship(...) \equiv$... sh!mkArrive(si,sc) ... let pos = hs? ... end sqr!ready ... sq!c ... let c' = qs? ... end ... $harbour(...) \equiv$... let mkArrive(s,c) = sh? ... hs!mkQuay(q) ... end $quay(...) \equiv$... if sqr?=ready then let c = sq? ... qs!c' ... end else ... end

type M

We sketch and explain the following specification text:

```
channel t_p,t_q : Bool, k : M
                                               variable w:M;
value
                                               q:while t_q? do
 P() \equiv
                                                 action_q1;
 p:while t_p? do
                                                 v := k ? in
   action_p1;
                                                 action_q3
   k ! v
   action_p3
  end
                                               R() \equiv P() \parallel Q()
```

• External events occur in an outside environment, "around" the part of the domain being considered — i.e., interfacing with it and are being communicated to that part. Or external events occur within the domain being considered, and are being communicated to "somewhere" outside the part of the domain being considered.

• Internal events occur in one part of the domain being considered and are destined for, i.e., communicated to, another part of the domain being considered — in which case we consider those parts as belonging to different behaviours.

In RSL we may model events in terms of RSL/CSP inputs/outputs:

type

```
ShipId, ShChar, HAPos, BuoyPos, QuayPos
MSG == mkArrive(shid:ShipId,shchar:ShChar)
    | mkHoldArea(p:Pos)
     mkBuoy(b:BuoyPos)
     | mkQuay(q:QuayPos) | ...
ArrDep == ready | depart
Cargo
```



• Behaviours

- Some phenomena (or concepts) are thought of as behaviours. They proceed, typically in time, by
 - ★ performing functions (actions),
 - \star generating or responding to events
 - \star and otherwise interacting (i.e., synchronising and communicating) with other behaviours.

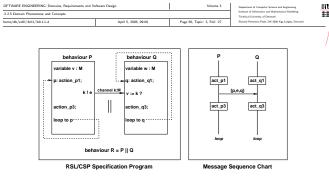


Figure 1.1: Informal process diagram

TWARE ENGINEERING: Domains, Requirements and Software Design

5 Domain Phenomena and Concepts

• If channels are nonpersistent, i.e., are 0-capacity buffers, then we say that the fact that the computation based on behaviour P does not proceed to its next action before the computation based on behaviour Q has consumed the message (sent by Q) constitutes a *synchronisation*.

- In either case, the message being transferred constitutes a communication.
- We shall usually use the term behaviour in favour of process.
- However, when a behaviour (or rather, a set of behaviours) is implemented by (i.e., exists inside) the computer, we shall also call it a process.

To the right in Fig. 1.1 we have shown an MSC (message sequence chart). The loop annotation is strictly speaking outside the proper MSC syntax.

We refer to Vol. 1, Chap. 21 for a thorough coverage of CSP and RSL/CSP. And we refer to Vol. 2, Chap. 13 for a thorough coverage of MSC.

4575 1770 Ew +45 4593 00

© Dines Bjørner, Fredoxej 11, DW-2040 Holte, Denna

E-mails: dirBinm.dtu.dk, bjorner@gmail.com, dines@bjorner.biz; UFE: www.imm.dtu.dk;

TWARE ENGINEERING: Domains, Requirements and	Software Design	Volume 3	Department of Computer Science and Engineering	DTU
.5 Domain Phenomena and Concepts			Institute of Informatics and Mathematical Modelling Technical University of Desmark	Ħ
и/db/volll/3ch1/3ch1-ii-d	April 5, 2006, 09:00	Page 59, Topic: 3, Foil: 30	Richard Petersens Plads, DK-2000 Kgs Lyngby, Denmark	

 An example railway function is: (v) the issuance of a ticket in return for the monies it costs. The function issue takes monies (Mo), from-station (Sn), to-station (Sn), date (Da), train number (Tn) and the state of all train reservations (TnRes) as arguments and delivers a ticket (Ticket) and an updated state of all train reservations as results.

```
type
Mo, Sn, Da, Tn, TnRes, Ticket
value
```

issue: Mo \times Sn \times Sn \times Da \times Tn \to TnRes \to TnRes \times Ticket

525 3720, Fax: +45 4593 007

© Dines Rigemer, Fredourij 11, DK-2940 Holte, Denni

E-maile: dollimm.dru.dk, bjorner@gmail.com, diese@bjorner.biz; UPL: www.imm.dru.dk

.5 Domain Phenomena and Concepts			Technical University of Denmark	##
ne/db/volll/3ch1/3ch1-ii-d	April 5, 2006, 09:00	Page 61, Topic: 3, Foil: 32	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	
trmo				
type				/
Sn, Train				
value				
train_ride: $Sn^* \rightarrow N \rightarrow T$	$\operatorname{Train} \to \operatorname{N} \times \operatorname{Train}$			
train_ride(snl)(net)(trn)	=			
if len snl ≤ 1				
then				
(net,trn)				
else				
let (net',trn') = ge	t_on_train(hd snl)(net	t)(trn);		
let (net'', trn'') = tr	$rain_dept(hd snl,hd t)$	l snl)(net')(trn');		
let (net'',trn''') = r	ide(hd snl,hd tl snl)	(net")(trn");		
let (net''',trn''') =	arriv_and_stop(hd tl :	snl)(net''')(trn''');		
$let (net^{mm}, trn^{mm}) =$	$get_off_train(\mathbf{hd}\ \mathbf{tl}\ sn$	nl)(net"")(trn"");		
$train_ride(tl snl)(r$	et"")(train"")			
end end end end er	ıd end			
				•

WARE ENGINEERING. Domains, Requirements and Software Design
Volum
Domain Phenomena and Concepts

Example 1.9 Railway Functions: We continue Example 1.8.

- Next we give a set of definitions of functions.
- \bullet These evolve around
 - ⋆ channels and
 - \star function definitions with synchronisation and communication between functions.
- We may then claim that this formalisation more properly describes a behaviour.
- We have tried to make the two formalisations, the above and the below, as similar as possible.

Example 1.8 Railway Entities, Functions, Events and Behaviours: Our example derives from railways.

Example railway entities are: (i) the railway net (N), (ii) its lines (L), (iii) its stations (S), (iv) the units (U) of the net into which it can be decomposed (linear, switches, crossovers, etc.), etc.

```
type
```

N, L, S, U

value

is_Linear, is_Switch, is_Crossover: U \rightarrow \mathbf{Bool}

-mail: delferenda.dx, bjornefignal.com, dieselbjornebiz: URL: www.inm.du.dx/*db

POWRE ENCONCERNOC, Domains, Requirements and Software Design

5 Domini Planomens and Concepts

April 5, 2006, 69:00 Page 60, Topic 2, Feb. 31

Kinker Present Pain, Dr. 2006, 100:00 Page 60, Topic 3, Feb. 31

- \bullet (vi) An example railway behaviour is:
 - \star passengers getting on a train, at a station platform;
 - \star the departure of the train from the station ;
 - \star the ride of the train down the line , including the acceleration and deceleration of the train ;
 - \star the arrival of the train at the next station, $% \left(1\right) =\left(1\right) +\left(1\right) +$
 - \star and the alighting of passengers.

45 4525 3720, Fax: +45 4593 0074

© Dines Rjørner, Fredovij 11, DK-2940 Holte, Dec

E-mails: deliferendra de, bjornerligmail.com, dieselbjorner biz; URL: www.imm.dra.de,

 OFTWARE ENGINEERING, Domains, Requirements and Software Design
 Volume 3

 2.2.5 Domain Pharomena and Concepts
 April 5, 2006, 00:00

 Peggs 62, Topic: 3, Foll. 33

Department of Computer Science and Engineering Institute of Informatics and Mathematical Modelling Technical University of Desmark Richard Potersons Plads, DK-2000 Kgs.Lyngby, Denmar

• That is, these functions are all to be thought of as executing instantaneously

 The above behaviour was expressed purely functionally, with references only to simple mathematical functions.

- instantaneously.
- \bullet So what is their temporal behaviour, one may very well ask?
- It is the set of sequences of actions and events denoted by the function definitions.
- Temporality is exhibited by orderings of these actions and events.
- One may, however, read the above formula as if each function took some not-further-specified time to execute, i.e., to be applied.
- Thus you may trick yourself into believing that the formulas prescribe a timed behaviour.

: +45 4525 3720, Fax: +45 4593 0074

© Dines Rjømer, Fredovrj 11, DK-2940 Holte, Dr

 $E-mail::de\Phi en du.di, hjoener Bynail.com, die er Bhjoener hiz; URL::www.inen.dtu.dk/"d$

© Dines Signmer, Fredoxy 11, DN-2840 Holte, Den

E-mails: del@mm.dtu.dk, bjorner@gmail.com, dines@bjorner.biz; UFE: www.inen.dtu.dk/"db

(2) Dines Rijemer, Fredovej 11, DN-284

E-maile debiene das de hiererbynal con, desoblijoner biz URL: www.inm.da.de/"d

WARE ENGINEERING Domains, Requirements and Software Design

Domain Pleasurement and Concepts

Domain Pleasurement and Concepts

Tracking Industry (Johns A. 1988)

(do),voll (John)-3-d April 5, 2006, 60-00 Page 67, Topic 3, Fob. 38 Richard Research Rich (Dr. 2008 Fig. Fob.)

Extend Powers Find, (Dr. 2008 Fig. Fob.)

- We shall treat the concepts of phenomena and concepts in a later lecture.
- Suffice it for now to justify the above remarks as follows.
 - \star Entities typically are manifest; that is, they exist in time and space.
 - \star Functions can be conceived through their effects, but cannot, in and by themselves, be observed.
 - \diamond Nobody has ever seen the number which we may represent by any of the numerals 7, vii, seven, IIIIII, III, etc.
 - * Even more so for behaviours: we may observe a progression of changing entities, effects of function applications and events; but we cannot "see" the behaviour, only conceive of it!
 - ★ The same is true for events.

5 3220, Fax: +45 4593 0074 © Dines Sigmen, Fredox j 11, DM-2840 Holte, Den

E-mails: doffenn.dtu.ds, hjorner@gmail.com, diest@hjorner.biz; URL: www.imm.dtu.di

Vision 5 Described Contracts Registrations are Sentence (August Contracts Registration Cont

- \star We expect a domain description to be the basis for constructing a major part of the requirements, namely that part which we shall call the $domain\ requirements.$
- \star And we expect a domain description to be a basis for what in other contexts than software engineering is known as business process reengineering.

1720, Fax: +45 4993 0074 (i) Dines Bijerner, Fredoxyl 11, DH-2840 Holte, D

TWARE ENGINEERSNC Domains, Requirements and Software Design

Volume 3

Opposites of Company Science and Engineery

Transier Descriptions as Bases for Domain Theories

April 5, 2005, 60:00

Page 71, Topic: 3, Folk-42

Richard Promose Plant (Pack)

- The engineering artifacts, that such engineers build, embody, so-to-speak, fragments of these theories.
- \bullet For the class of application domains that was characterised as being
 - * end-user,
 - \star public administration and institution oriented,
 - * as well as business and industry oriented,
- for that class of human-made universes of discourse we cannot refer to any such similar theories!

TWORE DISCINSTRANC, Commains, Respirements and Surfaces Design

Finding Expectations from Descriptions

Finding Expectations from Descriptions

Volume 3

Output Compare Command Expectation from Descriptions

Volume 3

Output Compare Command Expectation on Management and Surfaces and Surfaces April 5, 2006, 09.000

Page 68, Topic: 3, Feb. 30

Robert Processor Response Proc

Further Expectations from Domain Descriptions

- What else must we expect from a domain description?
- Although we shall review domain engineering in in the next lecture, and treat domain engineering in detail in later lectures we shall just mention a few things.
 - \star We expect a domain description to be readable and understandable by all stakeholders of the domain, i.e., by all those people who "populate" the domain.
 - *We expect a domain description to be the basis for learning about the domain, that is, for education about and training in the domain say, for such people as are being hired into a job in the domain, or for such people that need services offered by the domain.

525 3720, Fax: +45 4593 0074 © Dises Rjømer, Fredoxij 11, DK-2940 Holte

Domain Descriptions as Bases for Domain Theories

- Physicists have spent the last 400 years studying nature.
- Traditional engineering disciplines, such as
 - \star civil engineering,
 - ⋆ mechanical engineering,
 - \star chemical engineering,
 - \star electrical engineering, and
 - \star electronics engineering,
- all build on various theories of physics and chemistry.

ed 605 2720, Fax: +66 609 0076 © Close Olymor, Fredonj 11, DK-2000 Hebrs, Denmark E-mail: differendas. dt, kjennedgenal.com, dentibljener kir. UKL: www.inen.dx.dx/

227 Demain Descriptions as Basin for Domain Theories

227 Demain Descriptions as Basin for Domain Theories

Description as Basin for Domain Theories

Descri

- Isn't it about time that we develop theories, such as physicists have done, for respective application domains?
- \bullet This author thinks so.
- With the principles and techniques of domain engineering the student will be well prepared to help contribute research and development on such a theory.
- But to do it properly, the student needs to learn additional principles, formal techniques and related tools.

Halle, Demands Evalue definence du Lighter (Egrand Long, descriptions (Ed. 1983). USL: www.iren.du.ell/do +65.05.200, For +6.650.0004 © Dissell (Ed. 1983). Consulting (Ed. 1983). Cons

Volume 3

Openment of Companies (American Enginements and Software Design

Openment of Companies (American Enginements and Software Design

Openment of Companies (American and Administration and Administ

More on Domain Engineering

- \bullet We have briefly previewed some domain concepts,
- there are many more.
- For domain engineers
 - * to know how to proceed, what to do,
 - \star and how to do it, and do it professionally, with assurance,

it is important that they know

- * what domain engineering entails.
- \star In particular they must know
- \star what should be, and not be, in a domain description document,
- * that is, its parts and structure.

4525 3720, Fax: +45 4593 0074

© Dines Rijamer, Fredorej 11, DH-2840 Holte, Denni

b-maric distance.docde, spinner@grad.com, direct@pinner.bd; OHC www.inm.dba.dk/

WARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DTU
.1 The Machine			Institute of Informatics and Mathematical Modelling Technical University of Denmark	DTU
м/db/vollI/3ch1/3ch1-ё-г	April 5, 2006, 09:00	Page 75, Topic: 4, Foil: 2	Richard Petersent Plads, DK-2000 Kgs.Lyngby, Denmark	
The Machine				

• We introduce the term machine.

Characterisation 1.7 By *machine* we shall understand a combination of hardware and software that is the target for, or result of, computing systems development.

- The objective of writing down requirements is to prescribe desired properties of a machine:
- the software and the hardware on which the software resides.

1525 3720, Fax: +45 4593 0074

Dines Bjørner, Fredovej 11, DH-2040 Holte, Denn

E-maile: dollimm.dtu.dk, bjorner@gmail.com, dinec@bjorner.biz; UFL: www.imm.dtu.d

FWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DTU
.3 General			Institute of Informatics and Mathematical Modelling Technical University of Denmark	==
ne/dlb/vollI/3ch1/3ch1-ii-r	April 5, 2006, 09:00	Page 77, Topic: 4, Foil: 4	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	/

General

Characterisation 1.9 By requirements we mean a document which prescribes desired properties of a machine: what the machine shall (must, not should) offer of functions and behaviours, and what entities it shall maintain.

Characterisation 1.10 By a requirements prescription we mean the process — and the document which results from the process — of requirements capture, analysis and synthesis.

5 3720, Fax: +45 4593 0074

© Dines Rigener, Fredox | 11, DN-2840 Hobs, Denna

E-maile: doffinen.dtu.dk, bjorner@gmail.com, diner@bjorner.biz; UPE: www.imm.dtu.dk;

TWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DTU
.4 Different Kinds of Requirements			Institute of Informatics and Mathematical Modelling Technical University of Deumark	Ħ
ne/db/voll1/3ch1/3ch1-ii-r	April 5, 2006, 09:00	Page 79, Topic: 4, Foil: 6	Richard Petersens Plads, DK-2000 Kgc.Lyngby, Denmark	_
	Different Kinds	of Requirements	•	

- \bullet We see four different kinds of requirements:
 - * business process reengineering,
 - * domain requirements,
 - \star interface requirements, and
 - \star machine requirements.

TOPICE EAST Commiss, Requirements and Software Design

Volume 3

Superment Engineering

Sequence of Companies And Software Design

Requirements Engineering

spids /vell/(3843/3845-Fr

April 5,2006, 00:00

Page 74. Topic: 4, Foll. 1

Stand Present Plats, CG-280 Kg. Lepty, Demoks

Topic 4

Requirements Engineering

 In this lecture we shall give a brief characterisation of what we mean by requirements engineering.

25 3720, Fax: +45 4593 0074

Dines Mylmer, Fredoxy 11, DR-2000 Hotte, Dr

Prize administration, approximation, annoughment over well-mentalist, or

OFTWARE ENGINEERING: Domains, Requirements and St 2.3.2 The Machine Environment	oftware Design	Volume 3	Department of Computer Science and Engineering Institute of Informatics and Mathematical Modelling Technical University of Denmark	DTU
home/db/voll1/3ch1/3ch1-ii-r	April 5, 2006, 09:00	Page 76, Topic: 4, Foil: 3	Richard Petersens Plads, DK-2900 Kgs.Lyngby, Denmark	
				_//

The Machine Environment

 \bullet We introduce the concept of the environment of a machine.

Characterisation 1.8 By machine environment we shall understand the rest of the world. More specifically, we mean

- \bullet those parts of the world which interface to the machine:
- \bullet its users, whether humans or technology
- So the objective of writing down requirements is also to delineate,
- \bullet to decide upon and distinguish between
 - \star what is to "belong" to the machine, and
- * what is to "belong" to the environment.

(j) Limit Egenni, Princip Li, Lin-Lini, M

E-mails: dublimen.dts.dk, bjomer@gmail.com, diner@bjomer.biz; URL: www.imm.dts.dk/



Tichnical University of Domesuk

Richard Priesense Plade, Dic 2000 Kgt. Lynghy, Denmark

**Deering We

Characterisation 1.11 By requirements engineering we understand the engineering, that is, we understand the development of requirements prescriptions: from requirements prescription via the analysis of the requirements document itself, its validation with stakeholders and its possible theory development.

45 4525 3720, Fax: +45 4593

© Dises Rjørner, Fredoxej 11, DK-2840 Holte, De

E-mails: dibBinen.dts.dk, bjørnerBgmail.com, dieerBbjørner.bkz; URL: www.imm.dts.dk,

 OFTWARE ENGINEERING. Domains, Requirements and Software Design
 Volume 3

 2.3.4 Different Kinds of Requirements
 April 5, 2006, 09:00
 Fage 80, Topic: 4, Feb. 7

Technical University of Denmark Richard Protesters Plads, DK-2800 Kgs.Lyngby

- Conventionally the following terms are in circulation:
 - \star systems requirements, which approximately covers our overall requirements,
 - \star user requirements, which approximately covers our domain and interface requirements,
 - \star functional requirements, which approximately covers our domain and interface requirements and
 - \star non-functional requirements: approximately covers our machine requirements.

4525 3720, Fax: +45 4593 0074

© Dines Bijørner, Fredovej 11, DK-2940 Holte, Denmark

E-mails: del@mm.dtu.dk, bjorner@gmail.com, dines@bjorner.biz; UFE: www.inen.dtu.dk/"db

45 4525 3720, Fax: +45 4593 0074

O Province Code | 11 PM Mile Into Donors

E-maile debiene das de hiererbynal con, desoblijoner biz URL: www.inm.da.de/"d

More on Requirements Engineering

- We have briefly previewed some requirements concepts.
- There are many more.
- For the requirements engineer
 - * to know how to proceed, what to do,
 - \star and how to do it, and do it professionally, with assurance,

it is important that that engineer knows

- \star what requirements engineering entails, in particular:
- \star what should be, and not be, in a requirements prescription document: its parts and structure.

TWARE ENGINEERING: Domains, Requirements and So	ftware Design	Volume 3	Department of Computer Science and Engineering	DTU
Software			Institute of Informatics and Mathematical Modelling Technical University of Denmark	==
ne/db/voll1/3ch1/3ch1-ii-s	April 5, 2006, 09:00	Page 83, Topic: 5, Foil: 2	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	
				-/

- In addition, as part of our wider concept of software, we also include
- a comprehensive collection of supporting documents:
 - * training manuals,
 - ★ installation manuals,
 - * user manuals.
 - * maintenance manuals, and
 - * development and maintenance logbooks.

So, software, as documentation, comprises many parts.

WARE ENGINEERING: Domains, Requirements and Software Design Software Design		Volume 3	Department of Computer Science and Engineering Institute of Informatics and Mathematical Modelling Technical University of Denmark	₩.
m/db/voll1/3ch1/3ch1-ii-s	April 5, 2006, 09:00	Page 85, Topic: 5, Foil: 4	Richard Patersens Plads, DK-2000 Kgs Lyngby, Dwanark	
• We make a dist	tinction between	two kinds of a	bstract software	

- specifications, and hence their designs:
 - \star the software architecture, \star and the component structure.
- After a brief presentation of these we shall comment on their nature.

TWARE ENGINEERING: Domains, Requirements and So	oftware Design	Volume 3	Department of Computer Science and Engineering	DTU
.1 Software Architecture and Software Architecture Desi	gn		Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
нe/dlb/vollI/3ch1/3ch1-ё-я	April 5, 2006, 09:00	Page 87, Topic: 5, Foil: 6	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	

Characterisation 1.15 By a software architecture design we mean the development process of going

- from existing requirements
- and possibly some already designed components
- to the software architecture -

producing all appropriate architecture documentation.

Topic 5 Software

Characterisation 1.12 By software we understand

- not only code that may be the basis for executions by a computer,
- but also its full development documentation:
 - ★ the stages and steps of application domain description,
 - * the stages and steps of requirements prescription,
- \star and the stages and steps of software design prior to code, with all of the above including all validation and verification

(including test) documents.

FTWARE ENGINEERING: Domains, Requirements and Sc .5 Software Design	oftware Design	Volume 3	Department of Computer Science and Engineering leatings of Informatics and Mathematical Modelling Technical University of Demands	DTU ##	ĺ
ome/db/vollI/3ch1/3ch1-ii-s	April 5, 2006, 09:00	Page 84, Topic: 5, Foil: 3	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark		ı
	Coftwome	Dagiam		_/	l

From an understanding of what software syntactically, i.e., as documents, "is", we can go on to characterise pragmatic and semantic aspects related to software.

Characterisation 1.13 By software design we understand

- the process,
- as well as all the documents resulting from the process,
- of turning requirements into executable code (and appropriate hardware)



Characterisation 1.14 By a software architecture we mean a first specification of software, after requirements,

- that indicates how the software is to handle the given requirements
- in terms of components and their interconnection -
- though without detailing, i.e., designing these components

Component Structure and Component Design

Characterisation 1.16 By a component structure we mean

• a second kind of specification of software — after requirements and software architecture

• one which indicates how the software is to implement individual components and modules

Characterisation 1.17 By component design (I) we mean the development process of going

- \bullet from existing requirements and a software architecture design
- to the detailed component modularisation -

producing all appropriate component and module documentation.

TWARE ENGINEERING: Domains, Requirements and So	ftware Design	Volume 3	Department of Computer Science and Engineering	DTU
.4 Modules, Components and Systems			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
ne/db/vollI/3ch1/3ch1-ii-s	April 5, 2006, 09:00	Page 91, Topic: 5, Foil: 10	Richard Petersens Plads, DK-2000 Kgs Lyngby, Denmark	/
	Modules, Compor	ents and System	s	

- The principle of grouping programming text into modules
- and collections of modules into components
- is both old (for modules since the late 1960s, e.g., Simula 67),
- and new (for components since the early 1990s).

Characterisation 1.18 By a module specification we shall understand a syntactic construct, i.e., a structure of program text, which, as a unit of program text, defines what we shall otherwise also call an abstract data type: namely a collection of data values and a collection of functions (i.e., operations) over these.

Modules, Components and Systems		Institute of Informatics and Mathematical Modelling Technical University of Desmank	
ıs/db/voll1/3ch1/3ch1-ii-s	April 5, 2006, 09:00	Page 93, Topic: 5, Foil: 12	Richard Petersens Plads, DK-2800 Kgs.Lyngby, Denmark
module m:			
types			
t1 = te1, t2 =	te2,, tt = tet	;	
variables			
v1 \mathbf{type} ta :=	ea, v 2 \mathbf{type} tb	:= eb,, vv t y	$\mathbf{ype} \text{ tc} := ec$
functions			
f1: $ti \rightarrow tj$, f1($ai) \equiv C_1(ai)$		
f2: $tk \rightarrow t\ell$, f2	$(ak) \equiv C_2(ak)$		
fn: $tp \rightarrow tq$, fr	$\alpha(ap) \equiv C_1(ap)$		
hide: fi, fj,, f	k		
end module			
Syntax sumantics and pragmatics: I			_

FWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DTU
.4 Modules, Components and Systems			Institute of Informatics and Mathematical Modelling Technical University of Desmark	Ħ
m/db/vollI/3ch1/3ch1-ii-s	April 5, 2006, 09:00	Page 95, Topic: 5, Foil: 14	Richard Petersent Plads, DK-2000 Kgs Lyngby, Denmark	_
	m			

Discussion 1.2 The idea is

- that a component specification to some surrounding text offers functions of some of its modules.
- The surrounding text may consist of modules, what we call initial modules of a software system.

Software Architecture Versus Component + Module Structure

- \bullet We are not saying that
 - \star one must first design the software architecture,
 - * and thereafter the component plus module structure.
- We are presently leaving their order of development and one of the two, or even a "mix" of them — unexplained!

Discussion 1.1 So, by an abstract data type, i.e., a module we mean a set of data values and a set of procedures (routines) that apply to such data values and yield such data values. Typically module specifications are of the following schematic form:

Characterisation 1.19 By a component specification we shall

• a set of type definitions,

usually understand

- a set of component local variable declarations,
- together defining a component local state,
- and a set of modules

The above is just a rough, generic characterisation of components.

• We may suggest a syntax for components:

component types: $\mathcal{T}_{i_1}, \mathcal{T}_{i_2}, \dots, \mathcal{T}_{i_t}$ variables: $\mathcal{V}_{j_1}, \mathcal{V}_{j_2}, \dots, \mathcal{V}_{j_v}$ modules: $\mathcal{M}_{k_1}, \mathcal{M}_{k_1}, \dots, \mathcal{M}_{k_m}$ hide: $\mathcal{H}_{\ell_1}, \mathcal{H}_{\ell_1}, \dots, \mathcal{H}_{\ell_h}$ end component

- T_i suggests some form of type definitions,
- V_i suggests some form of variable declarations,
- \mathcal{M}_k suggests some form of module specifications, and
- ullet Region suggests some form of export or hiding of module visible functions (etc.).

Systems, Design and Refinement

Characterisation 1.20 By a software system specification we shall understand

- a set of what we shall call initial modules
- together with a set of components -
- ullet and such that functions of the set of initial modules together invoke functions of modules in the set of components.

Systems are what we are developing.

TWARE ENGINEERING: Domains, Requirements and So	ftware Design	Volume 3	Department of Computer Science and Engineering	DTU
.5 Systems, Design and Refinement			Institute of Informatics and Mathematical Modelling Technical University of Denmark	==
ne/db/voll1/3ch1/3ch1-ii-s	April 5, 2006, 09:00	Page 99, Topic: 5, Foil: 18	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	
				-/

Characterisation 1.21 By software system design we understand

- the determination.
 - \star from domain requirements and from some interface requirements, of the software architecture, or
 - \star from machine requirements and from other interface requirements, of the component structuring plus initial modules.

Since software architecture design also entails determination of component structuring plus initial modules, we get, more generally, that software system design, in its first stage, i.e., where only the domain description and the requirements prescription exists, entails

TWARE ENGINEERING: Domains, Requirements and Sci	ftware Design	Volume 3	Department of Computer Science and Engineering	DTU
.5 Systems, Design and Refinement			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
н/db/vollI/3ch1/3ch1-ё-я	April 5, 2006, 09:00	Page 101, Topic: 5, Foil: 20	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	

Characterisation 1.22 By software system refinement we understand

- the stagewise and stepwise transformation of
 - ⋆ an abstract specification
 - \star into increasingly more concretely specified modules and components

TWARE ENGINEERING: Domains, Requirements and Sci	ftware Design	Volume 3	Department of Computer Science and Engineering	DT
.6 Components and Modules, Design and Refinement			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
ne/db/vollI/3ch1/3ch1-ii-s	April 5, 2006, 09:00	Page 103, Topic: 5, Foil: 22	Richard Pittersens Plads, DK-2000 Kgs.Lyngby, Denmark	
		D		

Components and Modules, Design and Refinement

Characterisation 1.25 By component design (II) we shall additionally understand

- the determination of which facilities,
 - * that is, which functions (defined locally "within" the component),
 - * and which types (defined globally, i.e., "outside"),
- the component shall offer.
- We shall also, by component design roughly speaking mean,
 - * the decomposition of the component into modules,
 - \star and hence, the functions offered by these modules

Discussion 1.3 The idea is that a system

- is a completely self-contained "item" of software,
- and that it is composed from components and the core, that is, the initial modules.

The idea is also

- \bullet that a most abstract level system may be the same as a software architecture.
- or a component plus initial module structure.

- the determination of the main (system) types of values.
- the determination of the basic structuring of and facilities (i.e., functions) offered by components, and
- the determination of such initial modules as are necessary to get the system executing once it is committed
- Usually a first stage of systems design is expressed abstractly,
- i.e., in a form not suited as a prescription for execution.
- Hence we need stages and steps of what is called refinement:

Characterisation 1.23 By abstract specification we mean one that indicates how requirements are to be implemented, but does it by using specification cum programming constructs that are not necessarily efficiently executable.

Characterisation 1.24 By concrete specification we mean one that uses specification cum programming constructs that prescribe efficient executions.

• One cannot expect a first attempt at component design to succeed

in finalising all aspects of an efficient implementation.

- As will be argued in the next lecture, separation of concerns makes it easier to tackle many diverse issues.
- Hence our development needs to proceed in stages and steps of refinement.

Characterisation 1.26 By component refinement we shall usually understand: • a concretisation of the usually initially abstractly defined

- component types,
- a concretisation of the usually initially abstractly specified initialisations of component variables, and, possibly,
- the refinement of the component modules

WARE ENGINEERING: Domains, Requirements and So	ftware Design	Volume 3	Department of Computer Science and Engineering	DTU
7 Code Design			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
s/db/voll1/3ch1/3ch1-ii-s	April 5, 2006, 09:00	Page 107, Topic: 5, Foil: 26	Richard Petersens Plads, DK-2000 Kgs Lyngby, Denmark	
	Code 1	Design		

- Finally, we reach the development stage where such program specifications are constructed that can be the basis for efficient execution.
- We call this kind of program specification 'code'.
- - ★ we shall assume the student to have a necessary background in programming
 - \star we shall not cover this topic in these lectures.

TWARE ENGINEERING: Domains, Requirements and 5	ioftware Design	Volume 3	Department of Computer Science and Engineering	DTU
.8 More on Software Design	Institute of Informatics and Mathematical Modelling Technical University of Denmark	==		
н/db/vollI/3ch1/3ch1-ё-я	April 5, 2006, 09:00	Page 109, Topic: 5, Foil: 28	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	

- That is neither the aim nor the objectives of these lectures.
 - * First, we have assumed some knowledge, education and training of the student.
 - \star Second we have to refer to special topic texts for detailed software design principles, techniques and tools.

TWARE ENGINEERING: Domains, Requirements and Sc	ftware Design	Volume 3	Department of Computer Science and Engineering	DTU
Discussion			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
нe/db/vollI/3ch1/3ch1-ё-я	April 5, 2006, 09:00	Page 111, Topic: 5, Foil: 30	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	
- We have not	as for arrange	t a d		

- We have not so far suggested
- \star similar structuring mechanisms
 - ♦ for domain descriptions,
 - or for requirements prescriptions.
- The software design cum programming language structuring constructs of components and modules, of types and variables, etc.,
 - * aid the developer in knowing "what to do next!"
 - ★ by providing documentation "standards".
- In the next lecture we shall preview such textual structuring (decomposition, composition) mechanisms for domain descriptions and requirements prescriptions.

Characterisation 1.27 By module refinement we understand:

• a concretisation of the usually initially abstractly defined module types.

- a concretisation of the usually initially abstractly specified initialisations of component variables and
- a concretisation of the usually initially abstract module function
- with the latter often entailing the introduction of additional auxiliary (i.e., hidden) function definitions

More on Software Design

- We have briefly previewed some software concepts.
- There are many more.
- We shall cover these and other software design concepts
 - * a little more in the next lecture,
 - * and in some detail in several later lectures.
- But, these lectures will not present anywhere near a fully satisfactory treatment of the software design problem.

Discussion

We have introduced the three main phases of software development:

- Domain engineering in which we describe "what there is'
- Requirements engineering in which we prescribe "what there shall be!"
- Software design in which we specify "how it will be!"
- We have indicated, assuming some programming maturity of the students, some software design structuring
 - * such as revolving around
 - components and modules (with locality and hiding of names),
 - types and variables (abstract and concrete), and
 - ♦ program statements and expressions summarised as clauses (C).

• Also, we have intimated, rather loosely,

- * notions of abstract versus concrete software design specifications,
- \star and hence we have intimated the entailed notion of refinement.
- We have not mentioned such stagewise and stepwise mechanisms,
- for domain descriptions and requirements prescriptions in general,
 - ★ other than for the business process reengineering, and domain, interface and machine requirements stages.
- Such development stage and step principles are mentioned already in the next lecture.

Topic 6

Phases, Stages and Steps of Development

- The terms phase, stage, and step, are just that: terms.
- They are meant to designate basically the same idea:
 - \star the decomposition of something occurring in time
 - * into adjacent, repeated or concurrent intervals.
- The "something" here is the development of software.
- The adjacent, concurrent or overlapping intervals
- are logically and otherwise distinguishable development activities.

TWARE ENGINEERING: Domains, Requirements and So	Powere Design	Volume 3	Department of Computer Science and Engineering	DTU
3 Linear, Cyclic and Parallel Development Activities			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
ne/db/yollI/3ch1/3ch1-iii	April 5, 2006, 09:00	Page 115, Topic: 6, Foil: 3	Richard Petersens Plads, DK-2800 Kgs.Lyngby, Desmark	
Linea	r. Cyclic and Paralle	el Development A	Activities	

- \bullet In the next lecture parts we shall present a view of the software development process as proceeding in strict linear order.
- Given human nature, such is rarely the case.
- At the end of this lecture we shall therefore present two additional views on the software development process:
 - ★ one in which iterations, backwards and forwards, are discussed;
 - * and one in which the concurrent tackling of logically separate stages or steps is discussed.

TWARE ENGINEERING: Domains, Requirements and Se	oftware Design	Volume 3	Department of Computer Science and Engineering	DTU	
Phases of Software Development	Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ			
ne/db/volll/3ch1/3ch1-iii	April 5, 2006, 09:00	Page 117, Topic: 6, Foil: 5	Richard Petersent Plads, DK-2000 Kgs Lyngby, Denmark	/	
				- /	

Phases of Software Development

- We have already introduced the main three phases of software development:
 - * domain development,
 - * requirements development, and
 - * software design.
- We have earlier argued for their distinctness, i.e., their focus on truly separate concerns,
- but we have also emphasised their desirable order, namely as listed.

FWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DI
Stages and Steps of Development			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ξ
н/db/volll/3ch1/3ch1-iii	April 5, 2006, 09:00	Page 119, Topic: 6, Foil: 7	Richard Petersens Plads, DK-2800 Kgs.Lyngby, Denmark	
- T4 :- 41 :	-+ +l+			

- It is thus important to also capture a notion of abstract versus concrete documentation — as here — of a phase of development.
- * The phase documentation can be more or less abstract, i.e., more or less concrete
- \star The phase documentation is abstract if primarily properties have been
- \star The phase documentation is concrete if primarily a model in terms of either a computable program, or a model in terms of such discrete mathematical notions as sets, Cartesians, lists, maps, etc., has been presented
- The distinction between stages and steps is basically a pragmatic distinction.
- That is, there is no "hard" theoretical basis for making that distinction, but there are good, sensible, practical reasons for doing so

A Principle of "Separation of Concerns"

- The main reason for decomposing the software development process into clearly distinguishable development activities
 - \star is to tackle separate development issues
 - * at separate times,
 - \star hopefully scheduling these in adjacent, concurrent or overlapping intervals
 - ⋆ in a fruitful, beneficial way.
- In the next parts of the lecture we shall briefly review possible decompositions.
- Each represent a concern; together they represent separation of concerns.

3.0.3 Linear, Cyclic and Parallel Development Activities	Institute of Informatics and Mathematical Modelling Technical University of Denmark			
home/db/vollI/3ch1/3ch1-iii	April 5, 2006, 09:00	Page 116, Topic: 6, Foil: 4	Richard Petersens Plads, DK-2800 Kgs.Lyngby, Denmark	
By a repeated, o	r cyclic, interval	we mean two	intervals, occurring	

- in non-overlapping time periods, in which basically the same item of work is done, i.e., repeated, for example, because a first iteration was not good enough.
- By concurrent or overlapping intervals we mean two (or more) intervals, in which clearly unrelated work items can be done, independently of one another, hence in parallel.

DFTWARE ENGINEERING: Domains, Requirements and So	Volume 3	Department of Computer Scie	
8.2 Stages and Steps of Development			Institute of Informatics and M Technical University of Denm
ome/db/volII/3ch1/3ch1-iii	April 5, 2006, 09:00	Page 118, Topic: 6, Foil: 6	Richard Petersens Plads, DK-
Stag	ges and Steps	of Developm	nent

- \bullet In order to capture a notion of development stage it is important to first capture a notion of the complete documentation — as here — of a phase of development.
 - * The documentation for a phase of development is complete
 - ⋆ if all there is to be documented at a certain level of abstraction has been so documented!
 - \star The idea of "all there is to be documented" is explained in a later lecture.

Characterisation 1.28 By a development stage we shall understand a set of development activities

- which either starts from nothing and results in a complete phase documentation
- or which starts from a complete phase documentation of stage kind, and results in a complete phase documentation of another stage kind

/vallI/3ch1/3ch1-iii	April 5, 2006, 09:00	Page 121, Topic: 6, Foil: 9	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	
Characterisation understand a way being comprehens as specifying (descriptions)	on 1.29 By a star of characterising ive (i.e., relatively cribing, prescribin such a way that of	age kind we sh a set of develor complete) an ag) a set of pro other such sets	hall loosely perment documents as d, at the same time, perties of what is of documents can be	_

TWANE ENGAINERING Domains, Requirements and Software Design

Volume 3

Dipolation of Compute George Governments

Stages and Stage of Development

Line of Indicators and Engineering

Line of Indicators and Engin

Example 1.10 $Stage\ Kinds$:

- Examples of domain stage kinds are:
 - \star (d₁) business processes,
 - \star (d₂) intrinsics,
 - \star (d₃) support technologies,
 - \star (d₄) management and organisation,
 - \star (d₅) rules and regulations and
 - \star (d_6) human behaviour.

Vision 2

Vision 3

Vision 3

Vision 3

Vision 4

Vision 5

Vision 5

Vision 5

Vision 5

Vision 6

Vision

- \bullet Examples of software design stage kinds are:
 - \star (s₁) software architecture,
 - \star (s₂) component design (in which the entire component structuring of a software architecture is decided),
 - \star (s_3) module design (in which all modules of all components are designed) and
 - \star (s_4) code.

1755, Far. +6 688 307 © Diess Rijners, Frankrij II, Dit 286 Nobe, Dennah Ennah didensellersk, Sjone-Appellion, deseMijners hir USI, was inn die

TWARE ENGINEERING: Domains, Requirements and Software Design Stages and Steps of Development				1	Volume 3	Department of Computer Science and Engineering	DT		
						Institute of Informatics and Mathematical Modelling Technical University of Denmark			
н/db/voll1/3ch1/3ch1-iii			 April 5, 2006, 09:00		Page	127, Topic: 6,	Foil: 15	Richard Petersens Plads, DK-2800 Kgs.Lyngby, Denmark	

- To properly describe what we shall wish to call a step of development it seems necessary to further elaborate on two concepts.
 - \star First the concept of a module of description.
 - We already covered a version of this notion in an earlier lecture, where it was "tied" to the concept of program specification (text).
 - We now enlarge upon the module concept and speak of similar, contained domain description and requirements prescription parts.

OFTWARE ENGINEERING. Comains, Requirements and Software Design Volume 3

3.2 Stages and Steps of Development
home (& brief Util Vish Jich Little 11 Acril 5, 2006, 09.00 Page 122, Topic, 6, Foil: 10

Characterisation 1.30 Thus a stage kind imposes an equivalence relation on a set of sets of related documents: some sets, s_k, s'_k, \ldots, s''_k as belonging to the same kind (k), other sets, $s_k, s'_{k'}, \ldots, s''_{k''}$ as belonging to different kinds (k, k', \ldots, k'') .

| FTEMBER ENGINEERING: Domains, Registrement and Software Dissign
| Volume 3 | Department Conjugation and Software Dissign | Volume 3 | Special Conjugation | Software Softwar

- \bullet Examples of requirements stage kinds are:
 - \star (r_1) business process reengineering,
 - \star (r_2) domain requirements,
 - \star (r_3) interface requirements and
 - \star (r_4) machine requirements.

3720, Fax: +45 4590 0074 (2) Direct Rijemer, Fredovej 11, DK-2040 Halte, S

 $\textbf{E-mail:} \ \, \textbf{definends.ds.} \ \, \textbf{bjornerBgmail.com.} \ \, \textbf{disentitioner-biz:} \ \, \textbf{URL:www.inen.dts.ds.} / \text{"d}$

3.2 Stages and Stages of Development

Nome/de/veili/2643/2643-iii April 5, 2006, 09.00 Page 126, Topic: 6, Folt: 14

 ${\bf Discussion~1.4~One~may~properly~argue~whether~the~following~are~not~also~stage~kinds~rather~than~steps:~domain~requirements }$

- (r_{2_1}) projection,
- (r_{2_2}) determination,
- \bullet (r_{2_3}) instantiation,
- \bullet (r_{2_4}) extension and
- \bullet (r_{25}) fitting.

It really is just a matter of convenience, hence pragmatics.

5 3220, Fax: +65 4593 0074 © Dines Rijemer, Fredovij 11, DK-2840 Halte, D

 OFTWARE ENGINEERING. Domains, Requirements and Software Design
 Volume 3

 3.2 Stages and Steps of Development
 April 5, 2006, 00.00
 Page 120, Topic 6, Feb. 16

Department of Computer Science and Engineering Institute of Informatics and Mathematical Modelling Technical University of Desmark Richard Petersons Plads, DN-2800 Kgs.Lyngby, Desmark

- \star Second we refer to the concept of *refinement*.
 - We also covered a version of this notion in an earlier lecture, where it was likewise "tied" to the concept of relation between pairs of program specifications (texts).
 - We now enlarge upon the refinement concept and speak of similar refinements of domain description modules as well as requirements prescription modules.

© Dinne Bijamer, Fredorij 11, DK-2000 Holte, Dennark E-maile: dellimm.dtu.ds, hjorner@gmail.com, dinne@hjorner.bis; URL: www.imm.d

O Day Day Color II W White Day of

E-maile dirbinunda. di, hjomerbgmail.com, diesebhjomer.hiz; URL: www.imm.dta.dk/"di

Characterisation 1.31 By a development step we mean a refinement of a description module, from a more abstract to a more concrete description.

- It may now be necessary to improve upon the characterisation of the concept of stage,
- so as to make the distinction between stages and steps
- more practical.

TWARE ENGINEERING: Domains, Requirements and Sci	ftware Design	Volume 3	Department of Computer Science and Engineering	DTU
.1 Stages of Domain Development	Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ		
w/db/vollI/3ch1/3ch1-iii	April 5, 2006, 09:00	Page 131, Topic: 6, Foil: 19	Richard Petersent Plads, DK-2000 Kgs.Lyngby, Denmark	- 1
		Development		/

- Within domain development we can distinguish the following major stages — on which work can beneficially be pursued basically in the order now listed:
 - * identification and classification of domain stakeholders, and
 - \star identification and modelling, relative to identified domain stakeholder classes, of a number of domain facets.

TWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DTU
.1 Stages of Domain Development			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
н/db/vollI/3ch1/3ch1-iii	April 5, 2006, 09:00	Page 133, Topic: 6, Foil: 21	Richard Petersens Plads, DK-2000 Kgs Lyngby, Denmark	

Characterisation 1.33 By a business process domain we shall understand

- one or more behavioural descriptions in which
- strategic, tactical and operational sequences of transactions of
- \bullet a business, an enterprise, a public administration, an infrastructure component.
- are given each from possibly a number of stakeholder perspectives

The business process facet overlaps with the next facets. So be it!

TWARE ENGINEERING: Domains, Requirements and Sc	oftware Design	Volume 3	Department of Computer Science and Engineering	DTU
.1 Stages of Domain Development	Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ		
ne/db/vollI/3ch1/3ch1-iii	April 5, 2006, 09:00	Page 135, Topic: 6, Foil: 23	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	

Characterisation 1.34 By domain intrinsics we shall understand

- those phenomena and concepts of a domain which are basic to any of the other facets (listed below),
- with such a domain intrinsics initially covering at least one specific, hence named, stakeholder view

Characterisation 1.32 By a development stage we mean a set of development activities

- such that some (one or more) activities have created new, externally conceivable (i.e., observable) properties of what is being described,
- whereas some (zero, one or more) other activities have refined previous properties

♦ These include: modelling business process facets of a domain,

- ♦ modelling intrinsics facets of a domain,
- ♦ modelling possible *support technology* facets of a domain,
- modelling possible management and organisation facets of a
- modelling possible rules and regulations facets of a domain,
- modelling possible script facets of a domain, and
- ♦ modelling possible human behaviour facets of a domain.

the travel (i.e., train journey); being ticketed and finishing the journey.

- We shall briefly characterise these stages shortly, and
- we cover them in detail in a much later lecture.

Example 1.11 Railway Business Processes: A simple business process is that of a passenger inquiring, with a travel agent, about train travel possibilities; being offered some alternatives; settling for one; reserving appropriate tickets; paying and collecting these; starting

Example 1.12 Rail Intrinsics: Examples of rail intrinsics are: the rail net, the lines, and the stations — as seen from the passenger perspective — and the above plus the rail units (whether linear [including curved], points [switches], crossover, etc.), connectors (that allow units to be put together), etc. — as seen from the rail net signalling staff; and so on.

Characterisation 1.35 By domain support technology we shall

understand

• ways and means of implementing certain observed phenomena

Example 1.13 Railway Support Technologies: A rail unit switch

can be implemented in either of a number of support technologies: as operated purely by human power, as operated, from afar by mechanical wires, as operated electromechanically, or as operated electronically and electromechanically (say, in interlocking mode).

TWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DTU
.1 Stages of Domain Development		Institute of Informatics and Mathematical Technical University of Denmark	Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
н/db/vollI/3ch1/3ch1-iii	April 5, 2006, 09:00	Page 139, Topic: 6, Foil: 27	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	
• Domain manag	gement			

- - \star ensures that these decisions are passed on to the ("lower") levels of management, and to "floor" staff,
 - ★ makes sure that such orders, as they were, are indeed carried out,
 - ★ handles undesirable deviations in the carrying out of these orders cum decisions,
 - * and "backstops" complaints from lower management levels and from floor staff

I WARE ENGINEERING Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	UIU
.1 Stages of Domain Development			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
ne/db/vollI/3ch1/3ch1-iii	April 5, 2006, 09:00	Page 141, Topic: 6, Foil: 29	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	
				-/

Characterisation 1.37 By domain organisation we shall

- the structuring of management and nonmanagement staff levels, and
- \bullet the allocation of
 - * strategic.
 - \star tactical and
 - * operational

concerns to within management and nonmanagement staff levels.

TWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DTU
.1 Stages of Domain Development			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
ne/db/voll1/3ch1/3ch1-iii	April 5, 2006, 09:00	Page 143, Topic: 6, Foil: 31	Richard Petersess Plads, DK-2000 Kgs.Lyngby, Denmark	_
Example 1 15 l	Railway Organis	ation: Examp	le 1 14 considered	

management functions. The number and specialised nature of these usually warrants corresponding organisational structures: executive management entrusted with strategic issues, mid-level management with tactical issues and "floor" (or operational) management with operational issues.

Characterisation 1.36 By domain management we shall understand such people

- who determine, formulate and thus set standards (rules and regulations, see next) concerning
 - * strategic,
 - \star tactical, and
 - * operational

decisions.

Example 1.14 Railway Management: An aspect of train operator management is that some functions, being of strategic nature, are considered on a yearly basis (whether to offer new train services). Other functions, being of a tactical nature, are considered more regularly, although not daily (whether prices should be lowered or raised due to lower, or higher costs, or due to competition or lack thereof). Yet other functions being of an operational nature, and are considered, and decided upon, "from hour to hour" (rescheduling trains due to delays, etc.).

- Hence we mean the "lines of command":
 - * who does what and
 - \star who reports to whom, both
 - ♦ administratively, and
 - ♦ functionally

Characterisation 1.38 By a domain rule we shall understand

• some text which prescribes how people or equipment are expected to behave when dispatching their duty, respectively when performing their functions

Example 1.16 Railway Rules: In China: At railway stations, no two (or more) trains are allowed to enter and/or leave, including basically move around, simultaneously. In fact, train arrivals and departures must be scheduled to occur with at least 2-minute intervals. Elsewhere: A line between neighbouring stations is usually segmented into blocks with the rule that at most one train may occupy any one block, or even, in cases, with at least one "empty" (i.e., no train block) between two trains.

FWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DTU
.1 Stages of Domain Development			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
н/db/vollI/3ch1/3ch1-iii	April 5, 2006, 09:00	Page 147, Topic: 6, Foil: 35	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denesark	- 1

Example 1.17 Railway Regulations: Regulations may thus prescribe properties that must hold when rescheduling trains, for instance, negotiating with neighbouring stations, etc. Or regulations may prescribe punitive staff actions when a train driver disobeys a train signal.

TOTAL ENGINEERING DOMAIN, (Vegatellands and Johnson Dengi)		Volum 3	Department of Computer Science and Engineering	DIO
1 Stages of Domain Development			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
ne/db/vollI/3ch1/3ch1-iii	April 5, 2006, 09:00	Page 149, Topic: 6, Foil: 37	Richard Petersens Plads, DK-2900 Kgs.Lyngby, Denmark	- 1
				-/

Example 1.18 Railway Staff Behaviour: A railway ticket collector may check and double-check that all passengers have been duly ticketed, or may fail to do so, or may deliberately skip checking a whole carriage, etc.

TWARE ENGINEERING: Domains, Requirements and So	ftware Design	Volume 3	Department of Computer Science and Engineering	DTU
Requirements Development			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
ne/db/vollI/3ch1/3ch1-iv	April 5, 2006, 09:00	Page 151, Topic: 7, Foil: 1	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	_
	Ton	io 7		

Requirements Development

• From an earlier lecture we repeat some of the below characterisations.

Characterisation 1.41 By requirements we shall understand a document which prescribes the desired properties of a machine: what the machine shall (must, not should) offer of functions and behaviours, and what entities it shall "maintain".

Characterisation 1.39 By a domain regulation we shall

 \bullet some text which prescribes what remedial actions are to be taken

when it is decided that a rule has not been followed according to its intention

understand

Characterisation 1.40 By domain human behaviour we shall understand

- any of a quality spectrum of humans carrying out assigned work:
 - * From careful, diligent and accurate,

- \star sloppy despatch and
- * delinquent work,

 \star outright criminal pursuit

Steps of Domain Development

- Steps of domain development are now to be seen as such activities
 - * which do not materially, i.e., in substance, change the properties of what is being described,
 - \star but which refine, from more abstract to more concrete, their way of description.
- \bullet Other than the above, we shall not cover the issue of domain development steps in the present lecture,
- but refer to much later lecture for more details.

Characterisation 1.42 By requirements prescription we mean the process — and the document which results from the process — of requirements capture, analysis and synthesis.

Characterisation 1.43 By requirements engineering we understand the development of requirements prescriptions: from requirements prescription via the analysis of the requirements document itself, its validation with stakeholders and its possible theory development.

• We see four different kinds of requirements:

* business process reengineering,

* domain requirements,

* interface requirements, and

* machine requirements.

POWE ENGINEERING Domains, Requirements and Software Design

1 Sugar of Requirements Development

1 April 5, 2006, 09:00 Page 150, Topic 7, Fell 5

Business Process Reengineering Requirements

Characterisation 1.44 By business process reengineering requirements we understand

- \bullet such requirements which express assumptions about
- \bullet the future, usually changed, business process behaviour of the environment of the machine
- \bullet as brought about by the introduction of computing

Characterisation 1.45 By domain requirements we understand

- such requirements, to software, which are expressed
- \bullet solely in terms of domain phenomena and concepts

120, Sar + 64 100 80 Cons Signer, Fraher II, DK-386 Nate, Conserva Exactly defined about disprossing all specifications, developments in UK, was seen about A/

WARE ENCINEERING. Domains, Requirements and Software Design

Under 3

Stages of Requirements Development

Stages of Requirements Development

April 5, 2006, 09.000 Page 159, Topic 7, Fol: 9

Robert Process Place (1998) Design Process

Business Process Reengineering and Domain Requirements

 So in setting out, initially, acquiring (eliciting, "extracting") requirements,

- the requirements engineer naturally starts "in" or "with" the domain.
- That is, asks questions, of or to the stakeholders, that eventually should lead to the formulation of business process reengineering and domain requirements.

OFT-WRE ENGINEERING Domains, Requirements and Software Design

Volume 3

2. A 1 Stages of Requirements Domains, Requirements and Software Design

Land Stages of Requirements Development

boms/dp/sell/18/24/26/1-i-i
Agril 5, 2006, 00:00

Page 154, Topic: 7, Foil: 4

5

Technical University of Denmark

Richard Petersens Plain, DK-2000 Kgs.Lyngby, I

• Conventionally the following terms are in circulation:

- \star functional requirements which approximately cover our domain requirements;
- \star user requirements which approximately cover our interface requirements;
- \star non-functional requirements which approximately cover some of our machine requirements; and
- \star system requirements which approximately cover some other of our machine requirements.

0, Fax: +46 4995 0074 (C) Direct Iglener, Fredorig 11, DR-2980 Halfs, D

prince administration, springeration, amongonitate, over was installed, to

FTNARE ENGINEERING: Domain, Regimentals and Software Design

Volume 3

Deportment of Compute Goods and Engineering

Letter of Software Computer Com

- We suggest five domain-to-business process reengineering operations — which will be covered in a later lecture:
 - \star introduction of some new and removal of some old support technologies,
 - \star introduction of some new and removal of some old management and organisation structures,
 - \star introduction of some new and removal of some old rules and regulations,
 - \star introduction of some new and removal of some old work practices (relating to $human\ behaviours),\$ and related $access\ rights$ (i.e., password authentication, authorisation), and
 - \star related scripts.

25 3720, Fax: +45 4593 0074 (2) Dinne Rijemer, Fredovij 11, DK-2840 Hol

E-maile: db@mm.dtu.dk, bjomm@gmail.com, dimm@bjomm.biz; URL: www.imm.dtu.dk/

14.1 Sugar of Regiments Development

| Sugar of Regiments Development | Sugar of Regiments Development | Sugar of Regiments Development | Sugar of Regiment | Sugar o

- covered in a later lecture:
 * domain projection,
- \star domain determination,
- * domain instantiation,
- \star domain extension and
- ⋆ domain fitting.

3720, Fax: +45 4593 0074 © Dines Rjømer, Fredoxý 11, DK-2840 H

E-mails: de@mm.dra.ds, bjorner@gmail.com, diese@bjorner.biz; URL: www.imm.dra.ds/"d

OF TWARE ENGINEERING: Consists, Requirements and Software Divises

Volume 3
3.4.1 Stages of Requirements Development

April 5, 2006, 09:00 Page 160, Topic: 7, Foll. 10

Department of Computer Science and Engineering Institute of Informatics and Mathematical Modelling Technical University of Desmark Richard Processes Plads, DK-2800 Kgs.Lyngby, Denmark

• Interface Requirements

 ${\bf Characterisation~1.46~By~interface~requirements~we~understand}$

- such requirements, to software, which are expressed
- \bullet in terms of domain phenomena shared between the environment and the machine

20, Fax: +65 4593 9074 (2) Dines Signerer, Fredoxej 11, DH-2860

E-maile: diellienen das de, bjorner@granil.com, diese@bjorner biz: UPE: www.inem.das.dk/"da

IS 3720, Fax: +45 4593 0074

© Dines Rjømer, Fredovij 11, DK-2840 Holte, Denmark

E-maile: dethins du de, bjonerbynal con, desoblyoner biz, URL: www.inen.du.de/"de

• We consider five kinds of interface requirements which will be covered in a later lecture: * shared data initialisation requirements, * shared data refreshment requirements,

* man-machine dialogue requirements,

 \star man-machine physiological interface requirements, and

 \star machine-machine dialogue requirements.

TWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DTU
1 Stages of Requirements Development			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
ne/dlb/vollI/3ch1/3ch1-iv	April 5, 2006, 09:00	Page 163, Topic: 7, Foil: 13	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	
				-/

• We shall, in particular, consider the following kinds of machine requirements — to be covered in a later lecture:

* performance requirements,

* dependability requirements,

* maintenance requirements,

 \star platform requirements and

 \star documentation requirements.

WARE ENGINEERING: Domains, Requirements and Software Design		Department of Computer Science and Engineering	DTU
Computing Systems Design		Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
April 5, 2006, 09:00	Page 165, Topic: 8, Foil: 1	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	
Ton	sic 81		-/
			Institute of Informatics and Mathematical Mobiling Tachnical University of Denmark

Computing Systems Design

- Given a comprehensive set of requirements, including, notably, machine requirements,
- one is then ready to tackle, systematically, the issue of implementing these requirements.
- Usually these requirements not only, as their main implication, direct us to design software,
- but also in many instances imply hardware design.
- In other words: computing systems design derives from requirements.

TWARE ENGINEERING: Domains, Requirements and So	oftware Design	Volume 3	Department of Computer Science and Engineering	DTU
.1 Stages and Steps of Hardware Design			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
и/db/vollI/3ch1/3ch1-v	April 5, 2006, 09:00	Page 167, Topic: 8, Foil: 3	Richael Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	

- Sometimes trade-off decisions have to be made as to whether a required function or behaviour is to be implemented in hardware or in software. These are called codesign decisions.
- Other than just mentioning these facts here, we shall not cover the subject till a much, much later lecture.

• Machine Requirements

Characterisation 1.47 By machine requirements we understand

- those requirements of software that are expressed
- primarily in terms of concepts of the machine

Steps of Requirements Development

- Steps of requirements development are now to be seen as such activities
 - * which do not materially, i.e., in substance, change the properties of what is being prescribed,
 - \star but which refine, from more abstract to more concrete, their way of prescription.
- \bullet Other than the above, we shall not cover the issue of requirements development steps in the present lecture,
- but refer to a much later lecture for more details.



Stages and Steps of Hardware Design

- Performance, dependability and platform requirements typically imply a need for rather direct considerations of hardware whether
 - \star computers, computer peripherals, or sensory and actuator technologies.
- By stages and steps of hardware design we thus mean such which
 - \star the overall composition of hardware: information technology units, buses, etc., and their interfaces, and
 - ⋆ the specific design of information technology units and buses, and so on.

Stages of Software Design

- We have briefly mentioned the problem before:
 - * sometimes a set of requirements and a set of (domain) assumptions on the stability of the environment and the execution platform
 - * allow us to first develop a high-level, i.e., abstract, software design from domain (and possibly some interface) requirements.
 - ★ At other times these assumptions are such (i.e., imply instability, such) that we must first, defensively, develop a less high-level, i.e., a less abstract, software design from machine requirements.

- \bullet In the former case we say that we are first designing a
 - \star software architecture: something that very directly reflects what the user most directly expects.
- In the latter case we say that we are first designing a
 - * software component and module structure: something that very directly reflects what some machine requirements imply.
- The boundary between the two design choices is not sharp.

.....

© Dines Bjørner, Fredoxy 11, DK-2010 Holte, Denma

E-maile: defilimm.dtu.dx, bjorner@gmail.com, dinee@bjorner.biz; URL: www.imm.dtu.dk/"

TWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DTU
.3 Steps of Software Design			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
ne/db/voll1/3ch1/3ch1-v	April 5, 2006, 09:00	Page 171, Topic: 8, Foil: 7	Richard Petersess Plads, DK-2000 Kgs.Lyngby, Denmark	
	Steps of Soft	tware Design		

- We have, for these lectures, assumed that the student already has some knowledge of programming, i.e., of software design, albeit at a perhaps rather concrete, i.e., coding, level.
- In line with this assumption we shall not treat the important concept of software refinement.

525 3720, Fax: +45 4593 00

Dines Bijamer, Fredoxij 11, DK-2940 Holte, Denma

E-maile: dellimm.dtu.dk, hjorner@gmail.com, dinee@hjorner.biz; UPE: www.imm.dtu.dk



- Ideally it would be nice if a software development could proceed linearly,
 - \star from domain development,
 - \star via requirements development,
 - \star to software design.

525 3720, Fax: +45 4593 00

© Dines Rigener, Fredox | 11, DN-2840 Hobs, Denna

E-maile: dofferendradis, bjorner@grail.com, diese@bjorner.biz; UPL: www.irem.dta.dk/"s

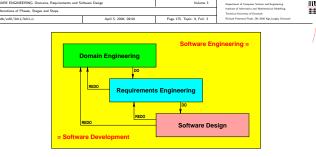


Figure 1.2: A diagramming of the iterative triptych phase development

- It is possible to identify other software design stages.
 - *Some may involve "conversion" from informal (or formal, abstract specification) language specifications to the identification and (hence) reuse of existing, "ready-made" and/or instantiatable (i.e., parameterisable) "off-the-shelf" (OTS) modules and components.
 - ⋆ Others involve conversion from informal (or formal, abstract specification) language specifications to formal (say, programming) language specifications, without the use of OTS software. This stage includes the final coding stage.
- \bullet Also here the boundaries are usually fuzzy.

20, Fax: +45 4540 0074

react around as a principal con, annual man, one was installed to

\$1.5 Says of Software Design

| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design
| Software Design

- such topics as:
 - \star Dijkstra's Discipline of Programming,
- \star Gries' Science of Programming,
- \star Reynolds' Craft of Programming,
- \star Hehner's Logic and Practical Theory of Programming,
- \star Jones' Systematic Software Development,
- \star Morgan's $Refinement\ Calculus$ or
- \star Back and von Wright's (earlier) Refinement Calculus.
- In a (much) later lectures we shall, however, briefly illustrate notions of software design refinement.

+45 4525 3720, Fax: +45 4583 0

© Dines Rjørner, Fredovij 11, DK-2940 Holte, Dec

E-maile: diellinen das die hjonnelligenal com, dinsellijoner hiz; URL: www.inen.das.dk

Department of Computer Science and Engineering Institute of Informatics and Mathematical Modelling Technical University of Denmark Richard Petersens Plads, DN-2800 Kgs.Lyngby, Denmark

- \bullet But reality seldom permits linear thinking and development.
- \bullet Instead one often encounters, in software developments which span the three phases, that they iterate:
- forwards and backwards between temporally neighbouring, even further temporally spaced phases.
- \bullet Figure 1.2 attempts to illustrate this iteration.

45 4525 3720, Fax: +45 4593

© Dines Rjørner, Fredoxej 11, DK-2840 Halte, Denr

 $\label{eq:conditional} \text{E-mail: distinct distance} \text{Distance} \text{Distance}$

 OFTWARE ENGINEERING Domains, Requirements and Software Design
 Volume 3

 3.6.2 Concurrency of Please, Stages and Steps
 April 5, 2006, 00:00
 Page 170, Topic: 0, Fol. 4

 homes (db)/vell/(2xh2)/2xh2-vir
 April 5, 2006, 00:00
 Page 170, Topic: 0, Fol. 4

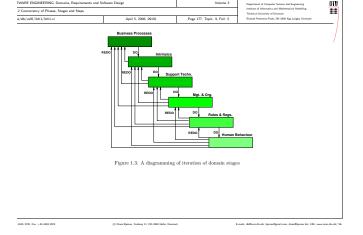
Concurrency of Phases, Stages and Steps

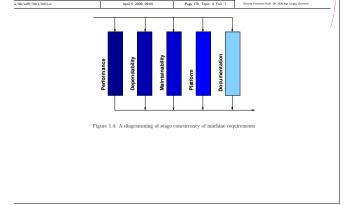
- Usually phases are developed, one at a time.
 - * First the domain phase is developed,
 - \star then the requirements phase, and
 - \star finally the software design phase.
- For stages of a phase and steps of different stages one may sometimes be able to carry out their development concurrently,
 - \star that is, by different teams of developers at the same,
 - \star or at least in partially overlapping time intervals.
- \bullet Stage of domain modelling usually follow the sequential order shown in Fig. 1.3.

E-maile dollarmatu de, bjorner@grail.com, diner@bjorner.biz; URL: www.inen.dtu.dk/"db

© Dines Rjømer, Fredoxej 11, DH-2840 Halte, Denmark

E-mails: detEmm.da.ds, bjomer@gnail.com, diesetBjomer.biz; URL: www.imm.da.ds/~ds





TWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DTU
The Concept of a Process Model			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
se/db/vollI/3ch1/3ch1-vii	April 5, 2006, 09:00	Page 181, Topic: 10, Foil: 2	Richael Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	

Characterisation 1.48 By a software development process model we shall understand

- a set of guidelines for how to start, conduct and end a software development project,
- a set of principles and techniques for decomposing these parts (start, conduct and end) into smaller, more manageable parts, and
- a set of principles, techniques and tools for what to do in, and how to do, these smaller parts
- In this lecture we shall thus very briefly summarise the basic ideas of the software development process model that these lectures are

based upon.

TWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3	Department of Computer Science and Engineering	DTU
The Triptych Process Model			Institute of Informatics and Mathematical Modelling Technical University of Denmark	Ħ
ne/dlb/vollI/3ch1/3ch1-vii	April 5, 2006, 09:00	Page 183, Topic: 10, Foil: 4	Richard Petersens Plads, DK-2000 Kgs.Lyngby, Denmark	
				-/

- In the next parts of the lecture we shall summarise the triptych process model.
- Throughout it is useful to keep in mind our remarks the previous lecture on iterative and concurrent phases, stages and steps of development.

• Typically * the domain requirements, \star the interface requirements and \star the machine requirements stages can be developed independently, i.e., concurrently, • Independent development is also appropriate for the individual "steps" within machine requirements: * performance. * dependability,

Topic 10 The Triptych Process Model A First View The Concept of a Process Model

- \bullet In software development many teams of many people each may have to collaborate
 - \star over long periods of time and

* maintainability, * platform, and * documentation requirements steps.

- \star over geographically widely distributed locations.
- It is therefore of utmost importance that clear guidelines, principles, techniques and tools are established and are agreed upon by all teams and people involved.
- \bullet The concept of a software development process model and its enunciation serves this role.



The Triptych Process Model

- Figure 1.2 highlighted what we shall refer to as the triptych phase process model.
- Figure 1.3 diagrammed an iterative process model for part of domain development.
- Figure 1.4 illustrates a concurrent process model for part of requirements development.

Topic 11 Conclusion to Chapter 1

It is time to complete this long introductory survey chapter. Summary

We have introduced crucial aspects of our approach to software development.

- Definitions of software engineering: First we brought in "old" and "new" definitions and characterisations of "what is software engineering"
- The triptych of software engineering: Then we surveyed the three key phases of our unique approach to software development: domain engineering, requirements engineering and software design.

TWARE ENGINEERING: Domains, Requirements and Software Design		Volume 3
Summary		
- (4b (Auril 6 2006 00:00	Base 185 Tenis: 11 Enil: 2

- Phases, stages and steps of software development: We reviewed these three phases and further suggested stages and steps of development within these phases and stages. We invite the reader to recapitulate the stages.
- Software development process models: We very briefly broached the topic of process models, in particular the one brought forward by this book. This process model will be enlarged upon in subsequent lectures.

.....

© Dines Bjørner, Fredoxej 11, DK-2940 Holte, Denmark

E-mails: db/Einm.dtu.dk, bjorner@gmail.com, dines@bjorner.biz; URL: www.imm.dtu.dk/"

DTU