
Lecture 1: Opening: Triptych & Documents, 1st Part

Dines Bjørner: 8th DRAFT: October 14, 2008

Introduction

- We put forward the triptych dogma.
- Overview the (three) triptych phases of software development,
- and their stages and steps of development.
- Then we overview the kind of documents
 - ★ being the target of
 - ★ and emanating fromthe three phases:
 - ★ the domain engineering,
 - ★ the requirements engineering, and
 - ★ the software designphases, namely respective

[Introduction]

- ★ informative,
- ★ descriptive / prescriptive / specificational documents.
- In this lecture we will cover some of the informative documents.

What Is a Domain ?

An Attempt at a Definition

Characterisation 1 (Domain) By a *domain* we shall understand

- a universe of discourse, small or large,
- a structure
 - ★ of entities, that is, of “things”, individuals, particulars
 - ◇ some of which are designated as state components;
 - ★ of functions, say over entities,
 - ◇ which when applied become possibly state-changing actions of the domain;
 - ★ of events,
 - ◇ possibly involving entities,
 - ◇ occurring in time and
 - ◇ expressible as predicates over single or pairs of (before/after) states; and
 - ★ of behaviours,
 - ◇ sets of possibly interrelated sequences of actions and events

[What Is a Domain ?]

Examples of Domains

- We give some examples of domains.
 - ★ (i) A country's railways form a domain
 - ◇ of the rail net with its rails, switches, signals, etc.;
 - ◇ of the trains travelling on the net, forming the train traffic;
 - ◇ of the potential and actual passengers, inquiring about train travels, booking tickets, actually travelling, etc.;
 - ◇ of the railway staff: management, schedulers, train drivers, cabin tower staff, etc.;
 - ◇ and so forth.

[What Is a Domain ?, Examples of Domains]

- ★ (ii) Banks, insurance companies, stock brokers, traders and stock exchanges, the credit card companies, etc., form the financial services industry domain.
- ★ (iii) consumers, retailers, wholesalers, producers and the supply chain form “the market” domain.
- ★ There are many domains
 - ◇ and the above have only exemplified “human made” domains,
 - ◇ not, for example, those of the natural sciences
- ★ We shall have more to say about this later.
- Essentially it is for domains like the ‘human made’ domains that these lectures will show you how to professionally develop the right software and where that software is right !

The Triptych Paradigm

- *Before software can be designed*
- *one must understand its requirements.*
- *Before requirements can be expressed*
- *one must understand the application domain.*

The Triptych Phases of Software Development

The Three Phases

- As a consequence of the “dogma”
- we view software development as ideally progressing in three *phases*:
 - ★ In the first phase, ‘Domain Engineering’, a model is built of the application domain.
 - ★ In the second phase, ‘Requirements Engineering’, a model is built of what the software should do (but not how it should that).
 - ★ In the third phase, ‘Software Design’, the code that is subject to executions on computers is designed.

[The Triptych Phases of Software Development]

Attempts at Definitions

Characterisation 2 (Domain Engineering) By *domain engineering* we shall understand

- the processes of constructing a domain model,
- that is, a model, a description, of the chosen domain,
- as it is, “out there”, in some reality,
- with no reference to requirements, let alone software

Characterisation 3 (Requirements Engineering) By *requirements engineering* we shall understand

- the processes of constructing a requirements model,
- that is, a model, a prescription, of the chosen requirements,
- as we would like them to be

[The Triptych Phases of Software Development, Attempts at Definitions]

Characterisation 4 (Software Design) By *software design* we shall understand

- the processes of constructing software,
- from high level, abstract (architectural) designs,
- via intermediate abstraction level component and module designs,
- to concrete level, “executable” code

Characterisation 5 (Model) By a *model* we shall understand

- a mathematical structure whose properties
- are those described, prescribed or design specified by
 - ★ a domain description,
 - ★ a requirements prescription, respectively
 - ★ a software design specification

[The Triptych Phases of Software Development]

Comments on The Three Phases

- The three phases are linked:
 - ★ the *requirements prescription* is “derived” from the *domain description*, and
 - ★ the *software design* is derived from the requirements prescription
- in such a way that we obtain a maximum trust in the software:
 - ★ that it meets customer expectations: that is, it is the right software, and
 - ★ that it is correct with respect to requirements: that is, the software is right.

[[The Triptych Phases of Software Development](#), [Comments on The Three Phases](#)]

Characterisation 6 (Phase of Software Development) ●

By a *phase of development* we shall understand

- ★ a set of development stages
- ★ which together accomplish
- ★ one of the three major development objectives:
 - ◇ a(n analysed, validated, verified) domain model,
 - ◇ a(n analysed, validated, verified) requirements model, or
 - ◇ a (verified) software design.

These three “tasks”: a domain model, a requirements model, and a software design will be defined below. ■

[[The Triptych Phases of Software Development](#), [Comments on The Three Phases](#)]

Characterisation 7 (Software Development) ● Collectively
the three phases are included

- when we say ‘software development’

• ■

Stages and Steps of Software Development

- We make distinctions between
 - ★ phases of development (i.e., the domain engineering, the requirements engineering and the software design phases),
 - ★ stages of development — within a phase, and
 - ★ steps of development — within a stage.

[Stages and Steps of Software Development]

Stages of Development**Characterisation 8 (Stage of Software Development)** • By

a *stage of development* we mean

- ★ a major set of logically strongly related development steps
- ★ which together solves a clearly defined development task

-
- We shall later define the stages of the major phases,
- and we shall then be rather loose as to what constitutes a development step.

[Stages and Steps of Software Development]

Steps of Development

Characterisation 9 (Step of Software Development) • By

a *step of development* we mean

- ★ iterations of development within a stage
- ★ such that the purpose of the iteration is
 - ◇ to improve the precision or
 - ◇ make the document resulting from the step reflect a more concrete description, prescription or specification

Development Documents

- All we do, really, as software developers, can be seen as a long sequence of
 - ★ documenting, i.e., producing, writing, documents
 - ★ alternating with
 - ★ thinking and reasoning about and presenting and discussing these documents to and with other people:
 - ◇ customers, clients and
 - ◇ colleagues.
 - ★ Among the last documents to be developed in this series are those of the executable code.

[Development Documents]

- In this section we shall take a look at the kind of documents
 - ★ that should result from the various phases, stages and steps of development,
 - ★ and for whose writing, i.e., as “input”, aside from other documents, we do all the thinking, reasoning, and discussing.
- For any of the three phases of development, one can distinguish three classes of documents:
 - ★ Informative Documents (Slide 26)
 - ★ Modelling Documents (Slide 119)
 - ★ Analysis Documents (Slide 124)

Informative Documents

- An informative document ‘informs’.
- An informative document is expressed in some national language.
- Informative documents serve as a link between developers, clients and possible external funding agencies:
 - ★ *“What is the project name ?”* Item 1
 - ★ *“When is the project carried out ?”* Item 1
 - ★ *“Who are the project partners ?”* Item 2
 - ★ *“Where is the project being done ?”* Item 2
 - ★ *“Why is the project being pursued ?”* Items 3(a)–3(b)
 - ★ *“What is the project all about ?”* Items 3(b)–3(g)
 - ★ *“How is the project being pursued ?”* Items 4–6

[Informative Documents]

- And many other such practicalities.
- Legal contracts can be seen as part of the informative documents.
- We shall list the various kinds of informative documents that are typical for domain and for requirements engineering.

An Enumeration of Informative Documents

- Instead of broadly informing about the aims and objectives
- of a development project
- we suggest a far more refined repertoire of information “tid-bits”.
- A listing of the sixteen names of these “tid-bits” hints at these:

[Development Documents, Informative Documents, An Enumeration of Informative Documents]

1. Project Name and Date Slide 30
2. Project Partners ('whom') and Place(s) ('where') from Slide 31
3. [Project: Background and Outlook]
 - (a) Current Situation from Slide 34
 - (b) Needs and Ideas from Slide 37
 - (c) Concepts and Facilities from Slide 42
 - (d) Scope and Span from Slide 45
 - (e) Assumptions and Dependencies from Slide 49
 - (f) Implicit/Derivative Goals from Slide 52
 - (g) Synopsis from Slide 56
4. [Project Plan]
 - (a) Software Development Graph from Slide 61
 - (b) Resource Allocation from Slide 71
 - (c) Budget Estimate from Slide 75
 - (d) Standards Compliance from Slide 76
5. Contracts and Design Briefs from Slide 85
6. Logbook from Slide 108

[Informative Documents, An Enumeration of Informative Documents]

- We shall now explain each of these kinds of informative documents.

[Informative Documents]

Project Name and Dates

The first information are those of

- Project Name: the name of the endeavour;
- Project Dates: the dates of the project.

[Informative Documents]

Project Partners and Places

The second information is that of

- **Project Partners:** who carries out the project.

Full partner (collaborator) details are (eventually) to be given:

- ★ **Client(s):** full names, addresses, and possibly names of contact persons, etc., of the people and/or companies and/or institutions who and which have ‘ordered’ the project and who and which shall receive its resulting documents.
- ★ **Developer(s):** full names, addresses, and possibly names of contact persons, etc., of the people and/or companies and/or institutions who and which are primarily developing the deliverables of the project and who and which shall receive its main funding.

[**Development Documents**, **Informative Documents**, **Project Partners and Places**]

- ★ **Project Consultant(s)**: full names, addresses, and possibly names of possible consultants, i.e., companies and/or individuals outside “the circle” of clients and developers.
- ★ **Project Funding Agencies**: full names, addresses, possibly names of contact persons, etc., of the people and/or agencies who and which are possibly [co-]funding the project.
- ★ **Project Audience**: full names, addresses, and possibly names of contact persons, etc., of the people and/or agencies who and which are possibly (also) interested in the project.

[[Informative Documents](#), [Project Partners and Places](#)]

- **Project Places:** where is the project carried out ?
 - ★ Full addresses:
 - ◇ visiting and postal mailing addresses and
 - ◇ electronic addresses.

[Informative Documents]

Current Situation

- Usually a domain engineering project is started for some reason.
 - ★ Either the developer or the client, or both, have only scant knowledge of the domain,
 - ★ or, when they have it is not written down but is “inside” the heads of some or most of their (i.e., developer or client) staff.

[Informative Documents, Current Situation]

- Similarly a requirements engineering project is started for some reason.
 - ★ A common reason is that of the current situation on the client side.
 - ◇ Either no IT is used but there is a need for some IT,
 - ◇ or current IT is outdated,
 - ◇ or new demands are made by owners, management or employees in general at the client, demands that “translate” into altered or new IT;
 - ★ or customers of the client may have similar expectations — of better e-service etc., from the client, i.e., their provider.

[Informative Documents, Current Situation]

- For a software design project
 - ★
 - ★
 - ★
- The ‘Current Situation’ document
 - ★ must outline this in succinct terms:
 - ★ say half to a full page.

[Informative Documents]

Needs and Ideas

Needs

- Usually the current situation is paraphrased,
 - ★ i.e., accentuated, by expressions of specific ‘needs’ for a domain description,
 - ★ or for a requirements prescription,
 - ★ or for a completed software design, i.e., for software.
- The need for a domain description could
 - ★ either be that it should form the basis for an orderly process of requirements development,
 - ★ or the basis for teaching and learning courses, say for new staff of the enterprise (of the domain),
 - ★ or both.

[Informative Documents, Needs and Ideas, Needs]

- The need for a requirements prescription could
 - ★ either be that it should form the basis for an orderly process of requirements development,
 - ★ or the basis for a tender, i.e., an offer to develop some software,
 - ★ or both.

[Informative Documents, Needs and Ideas, Needs]

- Usually can express needs while at the same time indicate how one might foresee an expressed need being possibly fulfilled, i.e., achieved.
- A need for a software design may be that it must be based on an existing requirements prescription.
- A need for a requirements prescription may be that it must be based on an existing domain description.
- A need for a domain description may be that it must be just informal, another need may be that it be both informal and formal.

Informative Documents, Needs and Ideas

Ideas

- One thing are the ‘needs’.
- Another thing are the ‘ideas’.
- If there are needs but no ideas, or if there is no need but ideas, then “forget it”: no reason to embark on a development !
- By *ideas* we mean that there are some substantial concepts that, when properly deployed, can lead to a believable development,
 - ★ whether of a domain description,
 - ★ of a requirements prescription, or
 - ★ of a software design.

[Informative Documents, Needs and Ideas, Ideas]

- By *domain ideas*
- we mean such concepts “upon” or “around” which one can build, one can model, a domain description.
- Examples will be given later (Slide 225)
- By *requirements ideas*
- we mean such concepts “upon” or “around” which one can build, one can model, a requirements prescription.
- Examples will be given later (Slide 505)
- By *software design ideas*
- we mean such concepts “upon” or “around” which one can build, one can model, a software design.
- Examples will be given later (Slide 637)

[Informative Documents]

Concepts and Facilities

- The pragmatics of the ‘concepts and facilities’ section
- is to — ever so briefly — inform all parties to the contract of
- which are the most important ideas of the subject domain of the contract.
- A facility is a physical phenomenon (here embodied, for example, in the form of software tools)
- while a concept is a mental construction (covering, usually some physical phenomena or concepts of these).

[Informative Documents, Concepts and Facilities]

- In the context of informing only about a domain description development project
 - ★ the concepts and facilities are intended,
 - ★ in the document section of that name,
 - ★ to be the most pertinent concepts and facilities
 - ★ on which the domain description should focus.

[Informative Documents, Concepts and Facilities]

- In the context of informing only about a requirements prescription development project
 - ★ the concepts and facilities are intended,
 - ★ in the document section of that name,
 - ★ to be the most pertinent concepts and facilities of the requirements prescription:
 - ◇ which are the novel ideas which the requirements should be based.

[Informative Documents]

Scope and Span

Characterisation 10 (Scope) By *scope* — in the context of informative software development documentation — we shall understand

- an outline of the broader setting of the problem, i.e., the universe of discourse at hand.

Characterisation 11 (Span) By a *span* — in the context of informative software development documentation — we shall understand

- an outline of the more specific area
- and the nature
- of the problem that need be tackled.

[Informative Documents, Scope and Span]

- Let us examine a few generic cases of scope/span determination.
 - ★ (i) “Pure” domain engineering scope and span:
 - ◇ By ‘ “pure” domain engineering’ we mean a project aimed at just producing a domain model.
 - ◇ In such a case the scope should typically be chosen as wide as possible,
 - ◇ while the span is a proper, but not too “small” subset of the scope.

[Informative Documents, Scope and Span]

- ★ (ii) Domain and requirements engineering scope and span:
 - ◇ By ‘domain and requirements engineering’ we mean a project
 - ◇ first aimed at producing a domain model
 - ◇ and then, from it, “derive” a requirements model.
 - ◇ In such a case the scope should typically be chosen to be comfortably wider
 - ◇ than the scope of the requirements part of the project.

[Informative Documents, Scope and Span]

- ★ (iii) Requirements engineering and software design scope and span:
 - ◇ By ‘requirements engineering and software design’ we mean a project
 - ◇ first aimed at producing a requirements model
 - ◇ and then, from it, “derive” a software design.
 - ◇ In such a case the scope and span part of
 - ◇ the requirements part of the project should be equal.
 - ◇ Software design projects have their scope and span being set by the requirements part of the project.

[Informative Documents]

Assumptions and Dependencies

- There are two kinds of assumptions and dependencies.
 - ★ One kind has to do with sources of knowledge.
 - ◇ For domain development there needs to be the sources from which the domain engineer can learn about and develop the domain description. We assume and depend on that.
 - ◇ For requirements development there needs to be a domain description as well as people from whom the requirements engineer can elicit the requirements and thus develop the requirements prescription. We assume and depend on that.
 - ◇ And for software design there needs to be a requirements prescription. We assume and depend on that.
 - ★ The other kind has to do with delineation of the domain.

[Informative Documents, Assumptions and Dependencies]

- Usually a domain description
- (one upon which we base our (domain) requirements)
- leaves out
- what we might call the “fringes” of the domain,
- i.e., the environment of that domain.
- To also describe those parts might simply “be too much”!
- That environment is simply judged too large, too unwieldy, to describe.

[Informative Documents, Assumptions and Dependencies]

- Yet, sooner or later, that environment will show up in the requirements prescription, if it is not already in the domain description.
- The requirements prescription eventually, thus, comes to depend — maybe not exactly crucially, but anyway — on events originating in the environment, or the ability of the computing system to dispose of some output to that environment.
- In the ‘assumptions and dependencies’ project document
 - ★ the project responsible must clearly express
 - ★ these assumptions and dependencies.

[Informative Documents]

Implicit/Derivative Goals

- Usually computing systems provide, or offer, a large number of entities, functionalities, events and behaviours,
- and it is those requirements we prescribe.
- But those entities, functionalities, events and behaviours really do not themselves reveal why they are or were prescribed.
- Usually their prescription serves “ulterior” goals which cannot be quantified in a way that indicates what the prescribed computing system should offer.

[Informative Documents, Implicit/Derivative Goals]

- Typical meta-goals are such as:
 - ★ *“Deployment of the computing system should result in greater profits for the company.”*
 - ★ *“Deployment of the computing system should result in the company attaining a larger market share for its products.”*
 - ★ *“Deployment of the computing system should result in fewer worker accidents.”*
 - ★ *“Deployment of the computing system should result in more satisfied customers (and staff).”*

[Informative Documents, Implicit/Derivative Goals]

- Other kinds of meta-goals are:
 - ★ “The existence of a domain description will have led or should lead to better understanding of the domain, hence to improved performance of domain staff trained in the domain based on such domain descriptions.”
 - ★ “The existence of a requirements prescription will have led or should lead to more appropriately targeted software.”

[Informative Documents, Implicit/Derivative Goals]

- In the ‘implicit/derivative goals’ project document
 - ★ the project responsible must clearly express
 - ★ these implicit/derivative goals.

[Informative Documents]

Synopsis

- The four sub-groups of informative document parts:
 - ★ current situation,
 - ★ needs and ideas,
 - ★ scope and span, and
 - ★ concepts and facilities,
- form an introductory “whole”
- that now need be “solidified”.
- They need to be brought together in a more coherent fashion — in what we shall call the
 - ★ synopsis document

[[Informative Documents](#), [Synopsis](#)]

Characterisation 12 (Synopsis) By a *synopsis*¹ — in the context of informative software development documentation — we shall understand the same as a resumé, a summary, that is,

- a comprehensive view, that is,
- an extract of a combination of current situation, needs and ideas, concepts, and scope and span documentation
- informing about a universe of discourse for which some development work is desired, for example:

¹<http://home/db/tseb/kap11/11.1.html>, comprehensive view, from *synopsis*: to be going to see together.

[Informative Documents, Synopsis]

- ★ the construction of a domain description,
- ★ or the construction of a requirements prescription based on an existing domain description, or both;
- ★ or the construction of a software design based on existing requirements prescription;
- ★ or both (requirements and software design),
- ★ or all (domain, requirements and software design);
- ★ or the first two (domain and requirements)



End of Lecture 1

Dines Bjørner: 8th DRAFT: October 14, 2008