
Lecture 2: Opening: Documents, 2nd Part and Much More !

[Informative Documents]

Software Development Graphs

- Development projects need be managed.
- This is true also for single person projects.
- Management of domain engineering projects must take into account
 - ★ that these are normally research projects:
 - ◇ little is objectively known about the domain before it is properly described;
 - ◇ hence one must be prepared for “unforeseen” resource usage.

[Informative Documents, Software Development Graphs]

- Software development graphs are a means of capturing,
 - ★ either beforehand,
 - ★ during,
 - ★ or after the projecthow that project
 - ★ is to be done,
 - ★ is being done,
 - ★ or was done,respectively !

Informative Documents, Software Development Graphs

Graphs

Characterisation 13 (Software Development Graph) • By a *software development graph* we shall *syntactically* understand a labelled graph

- ★ whose distinctly labelled *nodes* (*vertexes*) designate development activities (phases, stages or steps),
- ★ and whose distinctly labelled, directed *edges* (*arcs*) designate *precedence relations* between (node designated) activities.

[Informative Documents, Software Development Graphs, Graphs]

- Semantically a software development graph designate a set of project behaviour designators.
 - ★ A *project behaviour designator* is
 - ◇ a sequence of *phase, stage* or *step state designators* and *state transition designators*.
 - ◇ A *phase, stage* or *step state designator* is a node label such that the node is that of a phase part, or a stage part, or a step part of a software development graph.
 - ◇ A *state transition designator* is an edge label such that the edge is that of an edge of a development graph



Informative Documents, Software Development Graphs

A Conceptual Software Development Graph

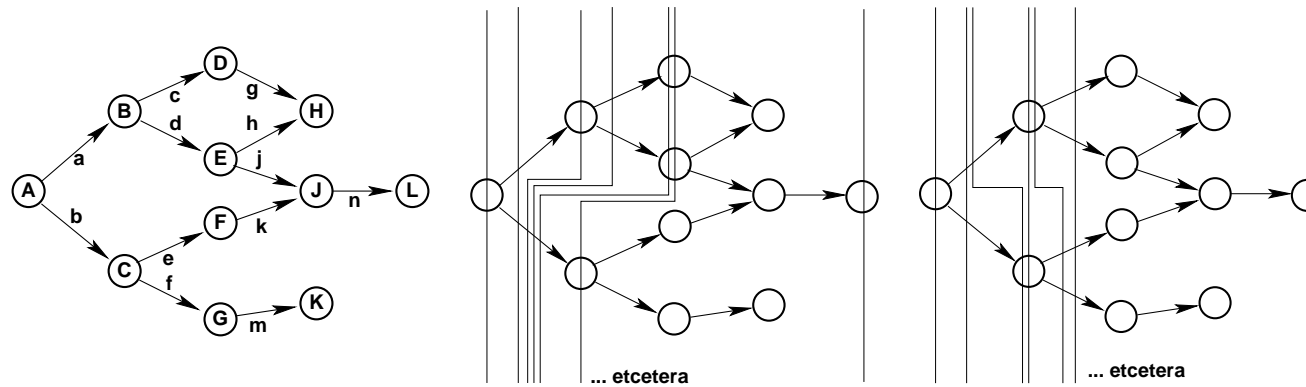


Figure 1.1: A software development graph (left) and two (incomplete) project behaviour designators (center and right)

- The above graph portrays the following incompletely listed project behaviour designator:

$$\langle \{A\}, \{a,b\}, \{B,b\}, \{c,d,b\}, \{D,E,b\}, \{D,E,C\}, \dots, \{L\} \rangle$$

[**Informative Documents**, **Software Development Graphs**, **A Conceptual Software Development Graph**]

- The “abstracted” software development graph of Fig. 1.1
- denotes a very large number of project behaviours,
- that is, a very large number of project behaviour designators,
- and, for each of these,
 - ★ depending on the states of phase, stage or step,
 - ★ as represented, for example, by the states of the documents related to each of the nodes,
- a very large number of (dynamic) behaviours.

Informative Documents, Software Development Graphs

Who Sets Up the Graphs ?

- Management is responsible for setting up an appropriate software development graph (for each project).
 - ★ A software development graph shows how management intends to pursue the project:
 - ◇ which phases, stages and steps to conduct,
 - ◇ that is, to which depth of adherence
 - ◇ to the triptych principles
 - ◇ management wishes to achieve its aims.

Informative Documents, Software Development Graphs

How Do Software Development Graphs Come About ?

- For a given, specific project, its software development graph comes about in any number of ways.
- (i) Either the project is a “repeat” project, that is, is developing a kind of software which has been developed before.
 - ★ In that case one simply uses the software development graph used in those earlier projects.
 - ★ But since there probably
 - ◇ are some small, or perhaps not even that small,
 - ◇ difference between the current project and the previous ones,
 - ◇ the currently chosen software development graph may be modified.
 - ★ Thus every software development graph will be recorded for possible re-use in future.
 - ★ It becomes part of the “corporate assets” of the software house.

[[Informative Documents](#), [Software Development Graphs](#), [How Do Software Development Graphs Come About ?](#)]

- (ii) Or the project is a “research” project, that is, is developing a new kind of software which has not been developed before.
 - ★ In that case one starts with the process diagram most appropriate for the project.
 - ◇ If it is a domain engineering project
 - then one starts with the domain engineering process graph of Fig. 2.4 on page 481 as the software development graph;
 - modifies this graph to suit the specific domain at hand,
 - all the while recalling that development of domain descriptions are really research rather than engineering tasks,
 - hence accepting that the software development graph need be modified along the way:
 - clear resource estimates of time and effort cannot be assured.

[Informative Documents, Software Development Graphs, How Do Software Development Graphs Come About ?]

- ◇ If it is a requirements engineering project
 - then one starts with the domain engineering process graph of Fig. 3.5 on page 610 as the software development graph;
 - and modifies this graph to suit the specific requirements at hand.
 - One must always be prepared to modify the software development graph along the way.

[Informative Documents]

Resource Allocation

Characterisation 14 (Software Development Graph Attribute)

An attributed software development graph

- is a software development graph whose nodes and edges have been assigned development attributes

[Informative Documents, Resource Allocation]

- Usually *node development attributes* include
 - ★ whether the node is a domain, a requirements or a software design development node;
 - ★ whether the node is a phase, stage, or step node;
 - ★ of what specific kind the node — when not just a phase node — is:
 - ◇ any one of the stages of the three triptych phases²;
 - ◇ any one of the 16 kinds of information document development steps enumerated on Page 28;
 - ◇ or any one of the many stages or steps of the domain and requirements modelling and analysis to be “revealed” in Chaps. and .

²The phases of domain and requirements modelling and analysis will first be “revealed” in Chaps. and — the only stage of the information document development is just

[Informative Documents, Resource Allocation]

- Given an attributed software development graph
 - ★ and given experience from projects “similar”
 - ★ to the one described by the graph
 - ★ one can now estimate resources to be allocated to each task,
 - ★ that is, to the carrying out the actions
 - ★ implied by each of its nodes.

[Informative Documents, Resource Allocation]

- These resource estimates are of the following kinds:
 - ★ number and qualifications of project staff;
 - ★ when, i.e., during which periods each individual, but not yet named staff, is to be available for the action denoted by the box being attributed;
 - ★ tools (office space, equipment (incl. IT equipment), software — by allocated staff members — to be available for that action;
 - ★ ‘begin’ and ‘end time’;
 - ★ etcetera.
- These estimates can be affixed to the nodes (boxes);
- thus augmenting its set of attributes.

[Informative Documents]

Budget (and Other) Estimates

- From the augmented (i.e., extended attributed) software development graph one can now derive a number of estimates:
 - ★ (i) a budget estimate, per phase and stage, and thus for the entire (software development graph [SDG] designated) project;
 - ★ (ii) a time estimate, per phase and stage, and thus for the entire (SDG designated) project;
 - ★ (iii) a staff estimate, per phase and stage, and thus for the entire (SDG designated) project (here it must be analysed which activities can occur in parallel) and usually in the form of a histogram;
 - ★ (iv) an equipment estimate, per phase and stage, and thus for the entire (SDG designated) project;
 - ★ etcetera.

[Informative Documents]

Standards Compliance

- A distinction is made between development standards and documentation standards.

Development Standards

- Usually development occurs in the context of following some development standards (one or more).
- The Institute of Electrical and Electronics Engineers (IEEE) has established a number of standards for the development of a various kinds of software.
- Other national and international organisations, including the International Organization for Standardization (ISO) and the International Telecommunication Union (ITU), have established similar standards.

Informative Documents, Standards Compliance

Documentation Standards

- Usually documentation occurs in the context of following some documentation standards (one or more).
- The Institute of Electrical and Electronics Engineers (IEEE) has established a number of standards also for the documentation of a various kinds of software.
- Other national and international organisations, including the International Organization for Standardization (ISO) and the International Telecommunications Union (ITU), have also established similar standards.

Informative Documents, Standards Compliance

Standards Versus Recommendations

- Some standards are binding, some are recommendations.
- Reference to specific standards and recommendations can be written into project contracts with the meaning that the project must comply with these standards and recommendations.
- Some standards mandate or recommend the use — and hence the documentation style — of certain development practices.
- Other standards mandate or recommend the use of specific spelling forms, mnemonics, abbreviations, etc.

Informative Documents, Standards Compliance

Specific Standards

- There are very many standards for software development and for its documentation.
- Some standards come and go. Others are quite stable.
- A study of more specialised standards reveals the following acronyms: MIL-STD-498, DOD-STD-2167A, RTCA/DO-178B, JSP188 and DEF STAN 05-91.
- The student is invited to search for these on the Internet.
- It therefore makes little sense for us to list other than a few clusters of seemingly more stable and trustworthy standards.

[Informative Documents, Standards Compliance, Specific Standards]

★ *International Organization for Standardization (ISO):*

<http://www.iso.ch/>

- ◇ ISO 9001: Quality Systems Model for quality assurance in design, development, production, installation and servicing
- ◇ ISO 9000-3: Guidelines for the application of ISO 9001 to the development, supply and maintenance of software
- ◇ ISO 12207: Software Life Cycle Processes
<http://www.12207.com/>

[Informative Documents, Standards Compliance, Specific Standards]

★ IEEE Standards: <http://standards.ieee.org/>

◇ IEEE Std 610.12-1990,

Standard Glossary of Software Engineering Terminology

This standard contains definitions for more than 1000 terms, establishing the basic vocabulary of software engineering.

◇ IEEE Std 1233-1996,

Guide for Developing System Requirements Specifications

This standard provides guidance for the development of a set of requirements that, when realized, will satisfy an expressed need.

[Informative Documents, Standards Compliance, Specific Standards]

- ◇ IEEE Std 1058.101987,
Standard for Software Project Management Plans
This standard specifies the format and contents of software project management plans.
- ◇ IEEE Std 1074.1-1995,
Guide for Developing Software Life Cycle Processes
This guide provides approaches to the implementation of IEEE Std 1074.
- ◇ IEEE Std 730.1-1995,
Guide for Software Quality Assurance Plans
The purpose of this guide is to identify approaches to good Software Quality Assurance practices in support of IEEE Std 730.

[Informative Documents, Standards Compliance, Specific Standards]

◇ IEEE Std 1008-1987 (reaffirmed 1993),
Standard for Software Unit Testing

The standard describes a testing process composed of a hierarchy of phases, activities, and tasks. Further, it defines a minimum set of tasks for each activity.

◇ IEEE Std 1063-1987 (reaffirmed 1993),
Standard for Software User Documentation

This standard provides minimum requirements for the structure and information content of user documentation.

◇ IEEE Std 1219-1992,
Standard for Software Maintenance

This standard defines a software maintenance process.

[Informative Documents, Standards Compliance, Specific Standards]

- ★ Software Engineering Institute (SEI): <http://www.sei.cmu.edu>
 - ◇ Software Process Improvement Models and Standards, including SEI's various Capability Maturity Models
- ★ *UK Ministry of Defence Standards* <http://www.dstan.mod.uk/>
 - ◇ 00-55: Requirements for Safety Related Software in Defence Equipment
<http://www.dstan.mod.uk/data/00/055/02000200.pdf>
 - ◇ 00-56: Safety Management Requirements for Defence Systems
<http://www.dstan.mod.uk/data/00/056/01000300.pdf>
- So, please, use the Internet for latest on standards relevant to your project.

[Informative Documents]

Contracts and Design Briefs

Contracts

- The

- ★ current situation,
- ★ needs and ideas,
- ★ concepts and facilities,

- ★ scope and span and
- ★ synopsis

document parts

- ★ set the stage for, and are a necessary background for contractual documents.

Characterisation 15 (Contract) By a *contract* — in the context of informative software development documentation — we shall understand a separate, clearly identifiable document

[Informative Documents, Contracts and Design Briefs, Contracts]

- which is legally binding in a court of law,
- which identifies parties to the contract,
- which describes what is being contracted for, possibly mutual deliveries, by dates, by contents, by quality, etc.,
- which details the specific development principles, techniques, tools and standards to be used and followed,
- which defines price and payment conditions for the deliverables,
- and which outlines what is going to happen if delivery of any one deliverable is not made on time, or does not have the desired contents, or does not have the desired quality, etc



[Informative Documents, Contracts and Design Briefs, Contracts]

- For national and for international contracts predefined forms which make more precise what the contracts must contain are usually available.
- We will not bring in an example.
- Such an example would have to reflect the almost ‘formal’ status of ‘legal binding’, and would thus have to be extensive and very carefully worded, hence rather long.
- Instead we refer to national and international contract forms.
- The software development field is undergoing dramatic improvements.
- Clients are entitled to have legally guaranteed quality standards (incl. correctness verification).

[Informative Documents, Contracts and Design Briefs, Contracts]

- Hence contracts will have to refer to
 - ★ the broader domain and give specific references to named domain stakeholders, if the development of a domain description is (to be) contracted; or
 - ★ existing domain descriptions and give specific references to named stakeholders, if the development of a requirements prescription is (to be) contracted; or
 - ★ existing requirements prescriptions and give specific references to named stakeholders, if the development of software is (to be) contracted.
- Therefore contracts should name “the methods” by means of which the deliveries will be developed — as we have indicated in bullet four of the characterisation.

Informative Documents, Contracts and Design Briefs

Contract Details**1. Overview:**

- Contracts between an organization and a software vendor should clearly describe the rights and responsibilities of the parties to the contract.
 - ★ The contracts should be in writing with sufficient detail to provide assurances for performance, source code accessibility, software and data security, and other important issues.
 - ★ Before management signs the contracts, it should submit them for legal counsel review.

[[Informative Documents](#), [Contracts and Design Briefs](#), [Contract Details](#), [Overview](#)]

- Organizations may encounter situations where software vendors cannot or will not agree to the terms an organization requests.
 - ★ Under these circumstances, organizations should determine if they are willing to accept or able to mitigate the risks of acquiring the software without the requested terms.
 - ★ If not, consideration of alternative vendors and software may be appropriate.

[Informative Documents, Contracts and Design Briefs, Contract Details]

2. General Issues of Licensing:

- Software
 - ★ is usually licensed, not purchased; and
 - ★ under licensing agreements, organizations obtain no ownership rights, even if the organization paid to have the software developed.
- In general, for domain descriptions and requirements prescriptions,
 - ★ a license should clearly define permitted users and sites.

[Informative Documents, Contracts and Design Briefs, Contract Details]

3. Copyright:

- Proprietary as well as open-source software
 - ★ are protected by copyright laws.
- If need be then clients and vendors
 - ★ must make sure that also their
 - ★ domain descriptions and requirements prescriptions
 - ★ are protected by being proprietary.

[Informative Documents, Contracts and Design Briefs, Contract Details]

4. Domain, Reqs. and Softw. Devt. Specs.:

- Contracts for the development of custom
 - ★ domain descriptions,
 - ★ requirements prescriptions, and
 - ★ software designmust be very specific about

[**Informative Documents**, **Contracts and Design Briefs**, **Contract Details**, **Domain**, **Reqs.** and **Softw. Devt. Specs.**]

- ★ the scope and span of domain descriptions and requirements prescriptions,
- ★ that requirements prescriptions build on accepted domain descriptions,
- ★ that requirements prescriptions are feasible and satisfiable,
- ★ and that software designs build on accepted requirements prescriptions.

[[Informative Documents](#), [Contracts and Design Briefs](#), [Contract Details](#)]

5. Performance Standards:

- This issue relates to requirements and software.
 - ★ When the requirements prescriptions are claimed feasible and satisfiable,
 - ◇ then there must be software that satisfies the requirements.
 - ◇ These requirements also include performance requirements,
 - ◇ part of the machine requirements to be covered later.

[Informative Documents, Contracts and Design Briefs, Contract Details]

6. Documentation, Modification, Updates and Conversion:

- A licensing or development agreement
 - ★ should require vendors to deliver appropriate documentation.
 - ★ This should include all kinds of documentation —
 - ★ such as defined later.
- A license or separate maintenance agreement
 - ★ should address the availability
 - ★ and cost of document updates and modifications.

[Informative Documents, Contracts and Design Briefs, Contract Details]

7. Bankruptcy:

- In addition to escrow agreements,
- organizations should consider the
 - ★ need for other clauses in licensing agreements
 - ★ to protect against the risk of a vendor bankruptcy.
- For mission-critical software,
 - ★ organizations should consult with their legal counsel
 - ★ on how best to deal with the Bankruptcy laws,
 - ★ which typically gives a bankrupt vendor discretion
 - ★ to determine which of its executory contracts it will continue to perform
 - ★ and which it will reject.
- Proper structuring of the contract
 - ★ can help an organization protect its interests if a vendor becomes insolvent.

[Informative Documents, Contracts and Design Briefs, Contract Details]

8. Regulatory Requirements:

- Domain descriptions, requirements prescriptions and software designs
 - ★ must individually often have to comply with
 - ◇ national (state and federal),
 - ◇ regional (NAFTA, EU, etc.),
 - ◇ and/or international (ICAO, IMO, etc.)regulatory agency requirements.
- These compliance requirements must be clearly stated in the contract.

[[Informative Documents](#), [Contracts and Design Briefs](#), [Contract Details](#)]

9. Payments:

- Software development contracts normally call for partial payments at specified milestones, with final payment due after completion of acceptance tests.
- Organizations should structure payment schedules so developers have incentives to complete the project quickly and properly.
- Properly defined milestones can break development projects into deliverable segments so an organization can monitor the developer's progress and identify potential problems.
- Contracts should detail all features and functions the delivered software will provide.
- If a vendor fails to meet any of its express requirements, organizations should retain the right to reject the tendered product and to withhold payment until the vendor meets all requirements.

[Informative Documents, Contracts and Design Briefs, Contract Details]

10. Representations and Warranties:

- Organizations should seek an express *representation and warranty* —
 - ★ this is a statement by which one party gives certain assurances
 - ★ to the other,
 - ★ and on which the other party may rely —in the document deliverables,
 - ★ that the licensed documentation
 - ★ whether a domain description
 - ★ a requirements prescriptions,
 - ★ or a software design (incl. code)does not infringe upon the intellectual property rights of any third parties.

[Informative Documents, Contracts and Design Briefs, Contract Details]

11. Dispute Resolution:

- Organizations should consider including dispute resolution provisions in contracts and licensing agreements.
 - ★ Such provisions enhance an organization's ability to resolve problems expeditiously
 - ★ and may provide for continued software development
 - ★ during a dispute resolution period.

[Informative Documents, Contracts and Design Briefs, Contract Details]

12. Agreement Modifications:

- Organizations should ensure software licenses clearly state
 - ★ that vendors cannot modify agreements
 - ★ without written signatures from both parties.
 - ★ This clause helps ensure there are no inadvertent modifications
 - ★ through less formal mechanisms some states may permit.

[Informative Documents, Contracts and Design Briefs, Contract Details]

13. Vendor Liability Limitations:

- Some vendors may propose contracts that contain clauses limiting their liability.
 - ★ They may add provisions that disclaim all express or implied warranties or that limit monetary damages to the value of the product itself, specific liquidated damages, etc..
 - ★ Generally, courts uphold these contractual limitations on liability in commercial settings unless they are unconscionable.
 - ★ Therefore, if organizations are considering contracts, they should consider whether the proposed damage limitation bears an adequate relationship to the amount of loss the financial organization might reasonably experience as a result of the vendor's failure to perform its obligations.
 - ★ Broad exculpatory clauses that limit a vendor's liability are a dangerous practice that could adversely affect the soundness of an organization because organizations could be injured and have no recourse.

[Informative Documents, Contracts and Design Briefs, Contract Details]

14. IT Security:

- We interpret this contract aspect only in the light of software. There is an ISO recommendation of IT Security:
 - ★ INTERNATIONAL ISO/IEC STANDARD 17799
 - ★ Reference number ISO/IEC 17799:2005(E),
 - ★ ISO/IEC 2005, ISO/IEC 17799:2005(E),
 - ★ Information technology, Security techniques: Code of practice for information security management,
 - ★ ISO copyright office, Case postale 56, CH-1211 Geneva 20, Switzerland. E-mail copyright@iso.org, Web www.iso.org. Published in Switzerland. Second edition, 2005-06-15.

We advice clients and developers to carefully adhere to that ISO recommendation.

[[Informative Documents](#), [Contracts and Design Briefs](#), [Contract Details](#)]

15. Subcontracting and Multiple Vendor Relationships:

- Some software vendors may contract third parties to develop software for their clients.
 - ★ To provide accountability, it may be beneficial for organizations to designate a primary contracting vendor.
 - ★ Organizations should include a provision specifying that the primary contracting vendor is responsible for the software regardless of which entity designed or developed the software.
 - ★ Organizations should also consider imposing notification and approval requirements regarding changes in vendor's significant subcontractors.

[Informative Documents, Contracts and Design Briefs, Contract Details]

16. Restrictions and Adverse Comments:

- Some software licenses include a provision prohibiting licensees from disclosing adverse information about the performance of the software to any third party.
 - ★ Such provisions could inhibit an organization's participation in user groups, which provide useful shared experience regarding software packages.
 - ★ Accordingly, organizations should resist these types of provisions.

Informative Documents, Contracts and Design Briefs

Design Briefs

Characterisation 16 (Design Brief) By a *design brief* we understand

- a clearly delineated subset text of the contract.
 - ★ This text (bullet three) describes what is being contracted for possibly mutual deliveries, by dates, by contents, by quality, etc., and
 - ★ (bullet four) it details the specific development principles, techniques and tools;
- that is, the design brief directs the developers, the providers of what the contract primarily designates,
- as to what, how and when to develop what is being contracted

[Informative Documents]

Logbook

Characterisation 17 (Logbook) By a *logbook* we understand

- a record, a set of notes,
- which as correctly as is humanly feasible,
- lists the
 - ★ development, release, installation, use, maintenance,
etc., history of a project

A logbook serves as a necessary reference in innumerable, usually unforeseeable instances of development.

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

Example 1 (Logbook) An “abstracted” ... (dot, dot, dot) example is:

2 Jan. 1991: Initial meeting between partners *ℰc*.

...

31 May 1993: Acceptance of domain model *ℰc*.

...

24 October 1994: Acceptance of requirements model *ℰc*.

...

3 June 1996: Acceptance of software delivery *ℰc*.

...

The *ℰc*. signify reports, and the ... signify other logbook entries. .



[Informative Documents]

Discussion of Informative Documentation

General

- We have identified some useful components of informative document parts.
- There may be other such informative parts. It all may depend on the universe of discourse, i.e., the problem at hand.
- We thus encourage the software developer to carefully reflect on which are the necessary and sufficient informative document parts.
- There is usually a separate set of informative documents to be worked out for each phase of development:
 - ★ the domain phase,
 - ★ the requirements phase, and
 - ★ the software design phase.

[Informative Documents, Discussion of Informative Documentation, General]

- The current situation, needs, ideas, concepts, scope, span, synopsis and contract document parts differ in content between these phases.
- Usually the informative document parts, although crucially important, need not require excessive resources to develop, but their development must still be very careful!
- In general, the informative document parts are concerned with the socioeconomic, even geopolitical, and hence pragmatic context of the projects about which they inform.
- As such they are “fluid”, i.e., less precise, in what they aim at and what their objectives are.
- The next two documentation kinds are, in that respect, much more precise, and much more focused.

Informative Documents, Discussion of Informative Documentation

Methodological Consequences: Principle, Techniques and Tools

Principle 1 (Information Document Construction) When first contemplating a new software development project,

- make sure — as the very first thing —
- to establish a proper complement of (all) informative documents.
- Throughout the entire development and after —
- during the entire lifetime of the result,
 - ★ whether a domain model,
 - ★ or a requirements model,
 - ★ or a software system —
- maintain this set of informative documents



[Informative Documents, Discussion of Informative Documentation, Methodological Consequences: Principle, Techniques and Tools]

Principle 2 (Information Documents) The informative documents must be authoritative,

- definitive and
- interesting to read

[[Informative Documents](#), [Discussion of Informative Documentation](#), [Methodological Consequences: Principle, Techniques and Tools](#)]

Technique 1 (Information Document Construction) First establish a document embodying the fullest possible table of contents, whether for

- just a
 - ★ domain development, or a
 - ★ requirements development, or a
 - ★ software design project,
- or for a combination of these.

[**Informative Documents**, **Discussion of Informative Documentation**, **Methodological Consequences: Principle, Techniques and Tools**]

Then fill in respective document parts,

- “little by little”, just a few sentences,
- using terse, precise, i.e., concise language,
- while avoiding descriptions (prescriptions and specifications) and analyses.

Throughout maintain clear monitoring and control of all versions of these documents. ■

[**Informative Documents**, **Discussion of Informative Documentation**, **Methodological Consequences: Principle, Techniques and Tools**]

Tool 1 (Information Document Construction) A text processing system, preferably L^AT_EX, but MS Word will do,

- with good cross-referencing facilities, even between separately ‘compilable’ documents,
- provides a ‘minimum’ tool of documentation.
- Add to this a reasonably capable version monitoring and control system (such as CVS) and you have a workable system

• ■

End of Lecture 2
