Introduction and Overview†

- Definitions.
- The general design process.
- A context for design: the waterfall model; reviews and documents.
- Some size factors.
- Quality and productivity factors.

---

†Material from: David Budgen (course text), and
Richard Fairley, *Software Engineering Concepts*, McGraw-Hill.

Definitions and descriptions

Software Engineering

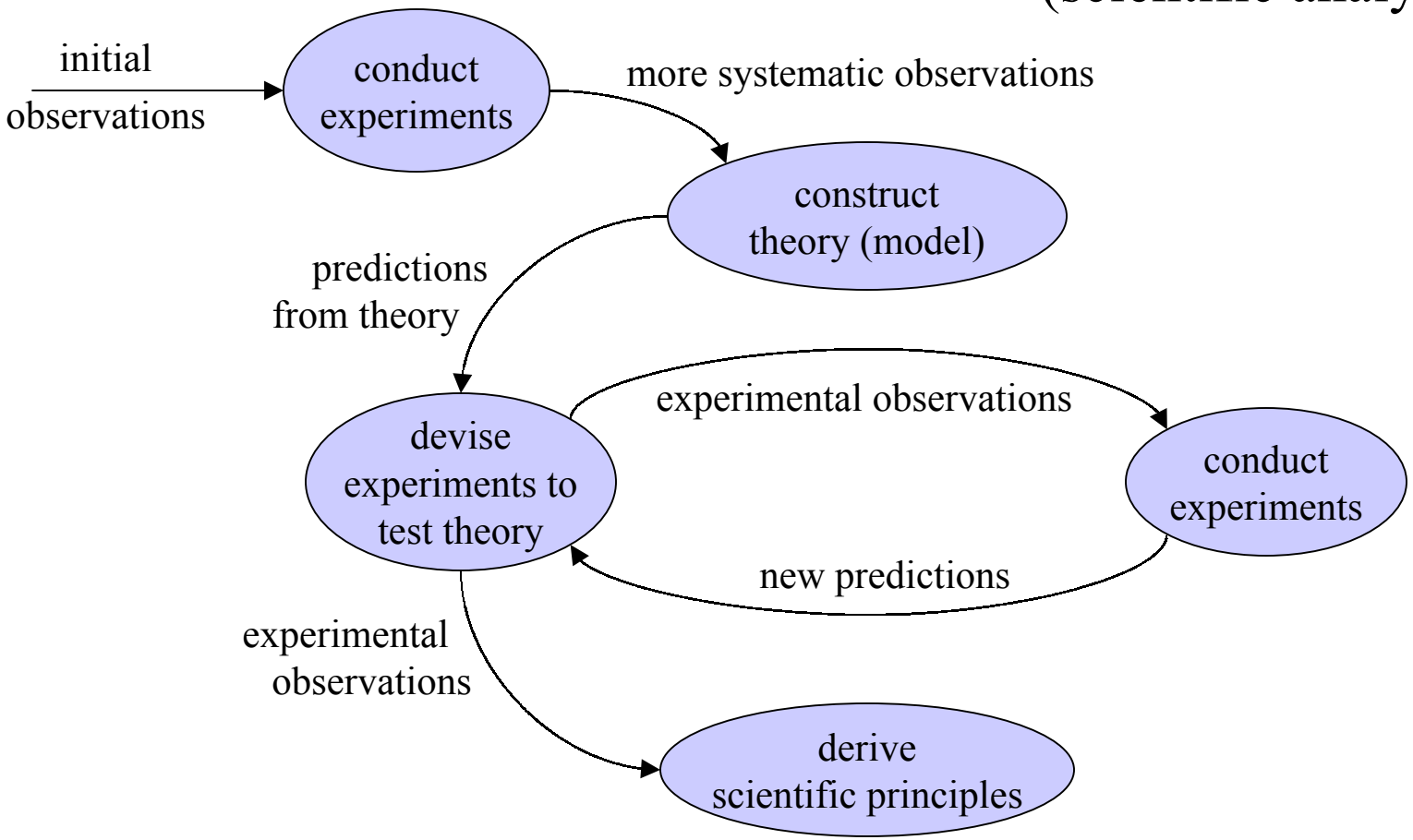
"*Software engineering is the technological and managerial discipline concerned with systematic production and maintenance of software products that are developed and modified on time and within cost estimates.*"

Software Design

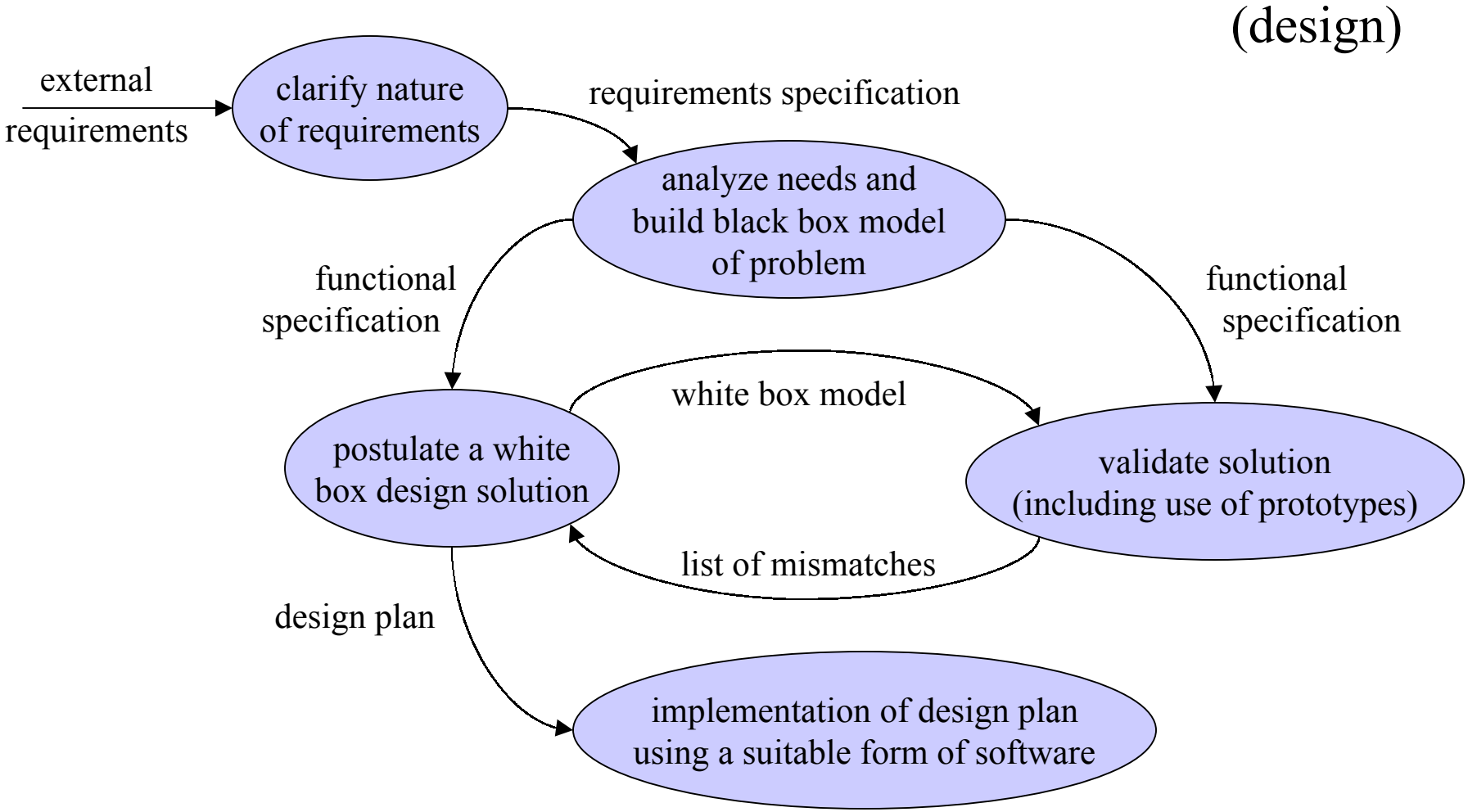
"*The fundamental problem is that designers are obliged to use current information to predict a future state that will not come about unless their predictions are correct. The final outcome of designing has to be assumed before the means of achieving it can be explored: the designers have to work backwards in time from an assumed effect upon the world to the beginning of a chain of events that will bring the effect about.*"

The general design process

(scientific analysis)



The general design process (cont'd)



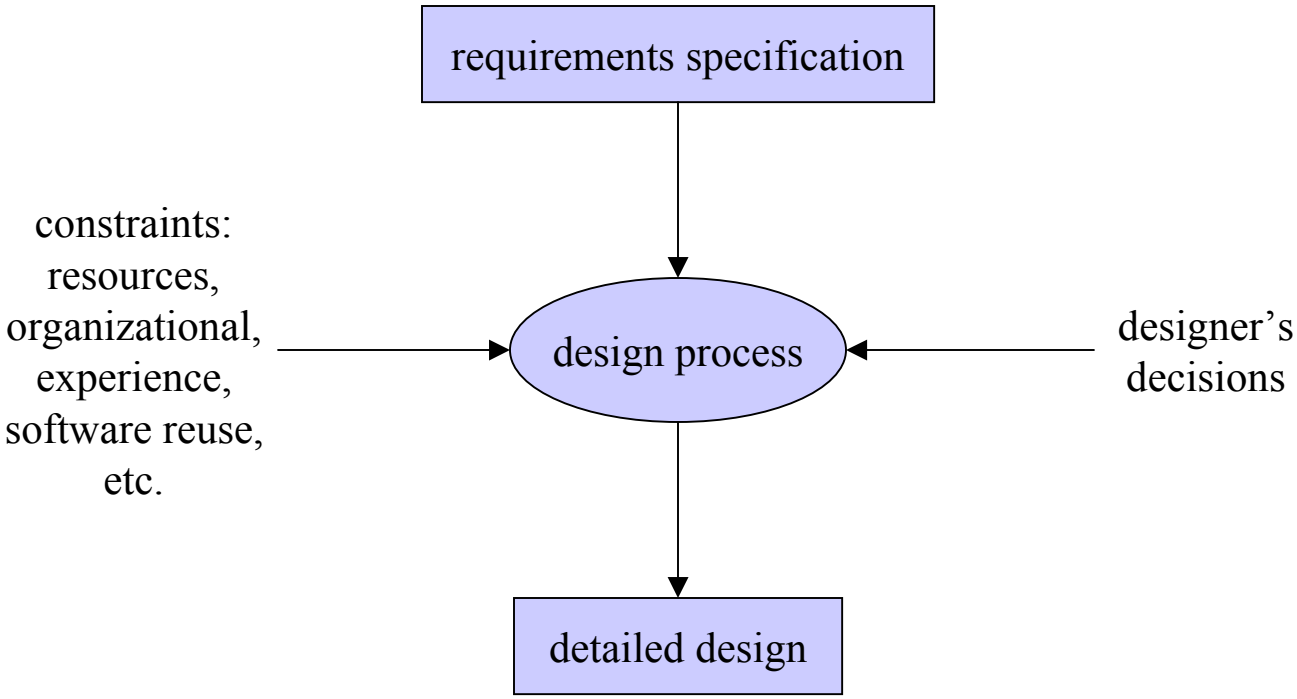
Universal factors

- Concepts.
- Languages (one or two dimensional).
- Tools.
- Process (or methodology, recipe, algorithm).

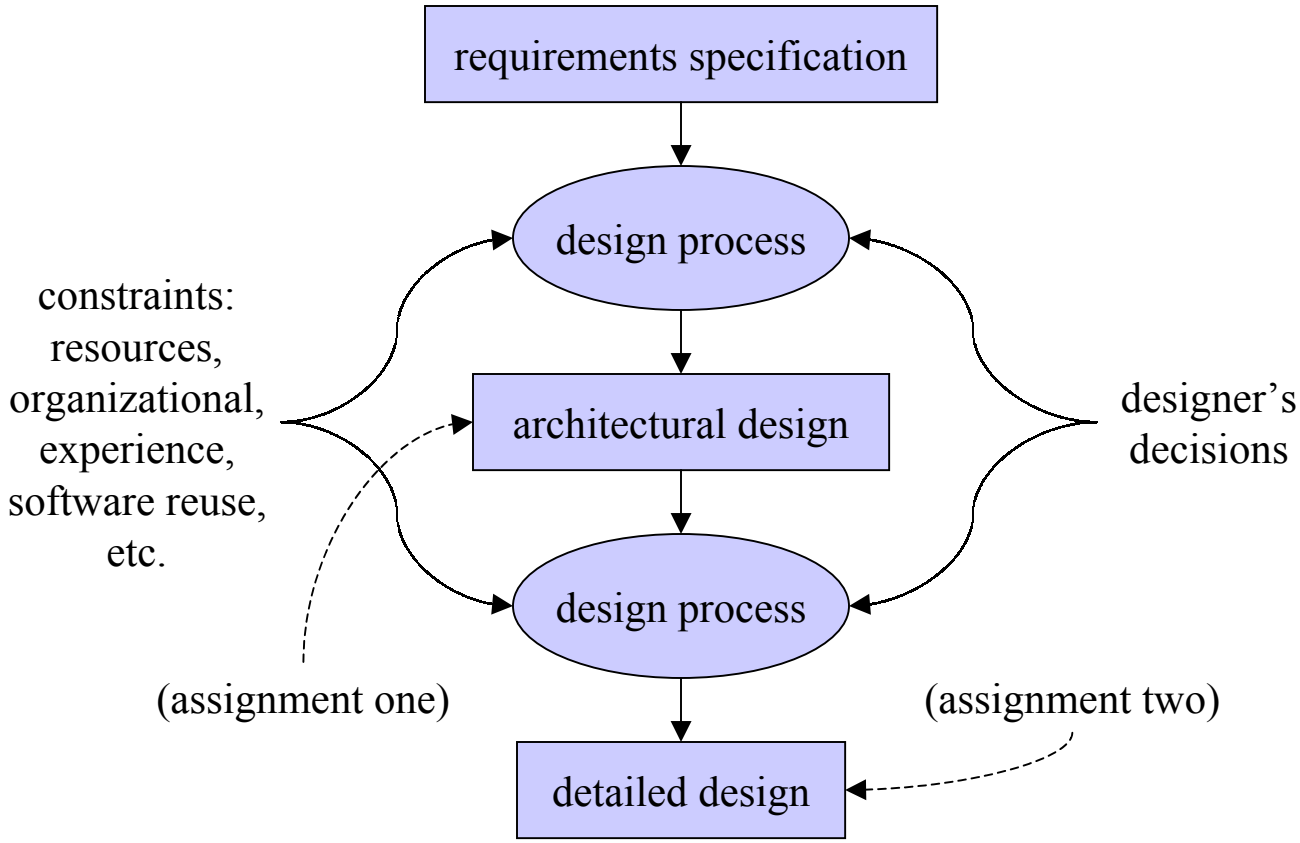
E.g.:

1. Building a house.
2. Building a compiler.

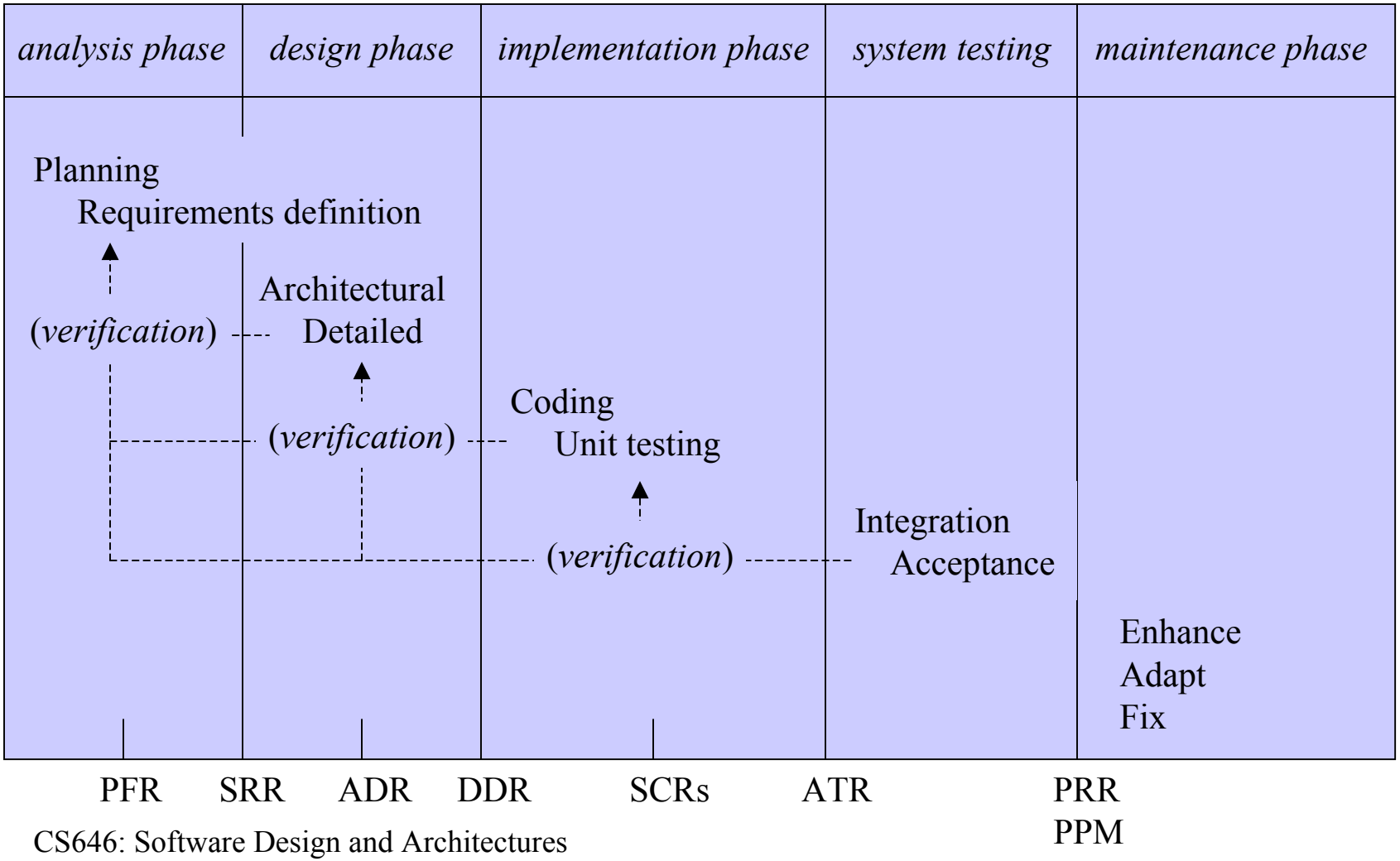
A general model of design



Separating architecture from detailed design



The waterfall life-cycle model



Reviews and documents

PFR (*product feasibility review*)	System Definition Project Plan
SRR (*software requirements review*)	Software Requirements Specification preliminary Verification Plan preliminary User's Manual
ADR (*architectural design review*)	Architectural Design Document
DDR (*detailed design review*)	Detailed Design Document Software Verification Plan User's Manual
SCR (*source code review*)	Walkthroughs and Inspections of the Source Code
ATR (*acceptance test review*)	Acceptance Test Plan
PRR (*production release review*)	(*all of the above*)
PPM (*project post-morten*)	Project Legacy

Contents of *system definition*

1. Problem definition.
2. System justification.
3. Goals for the system and the project.
4. Constraints on the system and the project.
5. Functions to be provided by hardware, software and people.
6. User characteristics.
7. Development, operating and maintenance environments.
8. Solution strategy.
9. Priorities for system features.
10. System acceptance criteria.
11. Sources of information.
12. Glossary of terms.

Contents of *project plan*

1. Life-cycle model (e.g., terminology, milestones and work products).
2. Organizational structure (i.e., management structure, team structure, work breakdown structure and statements of work).
3. Preliminary staffing and resource requirements, including staffing and resource schedule.
4. Preliminary development schedule (e.g., CPM graph and Gantt charts).
5. Preliminary cost estimate.
6. Project monitoring and control mechanisms.
7. Tools and techniques to be used.
8. Programming languages.
9. Testing requirements.
10. Supporting documents required.

Contents of *project plan* (cont'd)

11. Manner of demonstration and delivery.
12. Training schedule and materials.
13. Installation plan.
14. Maintenance considerations.
15. Method and time of delivery.
16. Method and time of payment.
17. Sources of information.

Contents of *software requirements specification*

1. Product overview and summary.
2. Development, operating and maintenance environments.
3. External interfaces and behavior (e.g., user display and report formats, user command summaries, high-level data flow diagrams, logical data sources and sinks, external data views and logical data dictionary)
4. Functional specifications.
5. Performance requirements.
6. Exception handling.
7. Early subsets and implementation priorities.
8. Foreseeable modifications and enhancements.
9. Acceptance criteria.
10. Design hints and guidelines.
11. Cross-reference index.

Contents of *architectural design document*

1. Data flow diagrams for the software product.
2. Conceptual specification of relevant data; specification of data sources.
3. Names, interface specifications and functional descriptions of subsystems and major modules.
4. Interconnection structure of the subsystems, major modules and data sources.
5. Timing constraints.
6. Exception conditions and handling.

Contents of *detailed design document*

1. Concrete internal design for relevant data.
2. Detailed algorithms.
3. Adaptations of existing code that will be reused.
4. Specific programming techniques required to solve unique problems.
5. Initialization procedures.
6. Detailed exception handling.
7. Task breakdown, including time estimates.
8. Integration test plan (e.g., test and schedule priority charts, CPM graphs and Gantt charts).

Contents of *verification plan*

1. Requirements to be verified.
2. Design verification plan.
3. Source-code test plan.
4. Test completion criteria.
5. Document verification plan.
6. Tools and techniques to be used.

Contents of *user's manual*

1. Introduction (i.e., product overview and rational, terminology and basic features, summary of display and report formats and outline of the manual).
2. Getting started (e.g., sign-on, help mode, sample run).
3. Modes of operation; commands, dialogues and reports.
4. Advanced features.
5. Command syntax and system options.

Contents of *acceptance test plan*

1. Requirements to be verified.
2. Test cases for each requirement.
3. Expected outcome of each test case.
4. Capabilities demonstrated by each test.

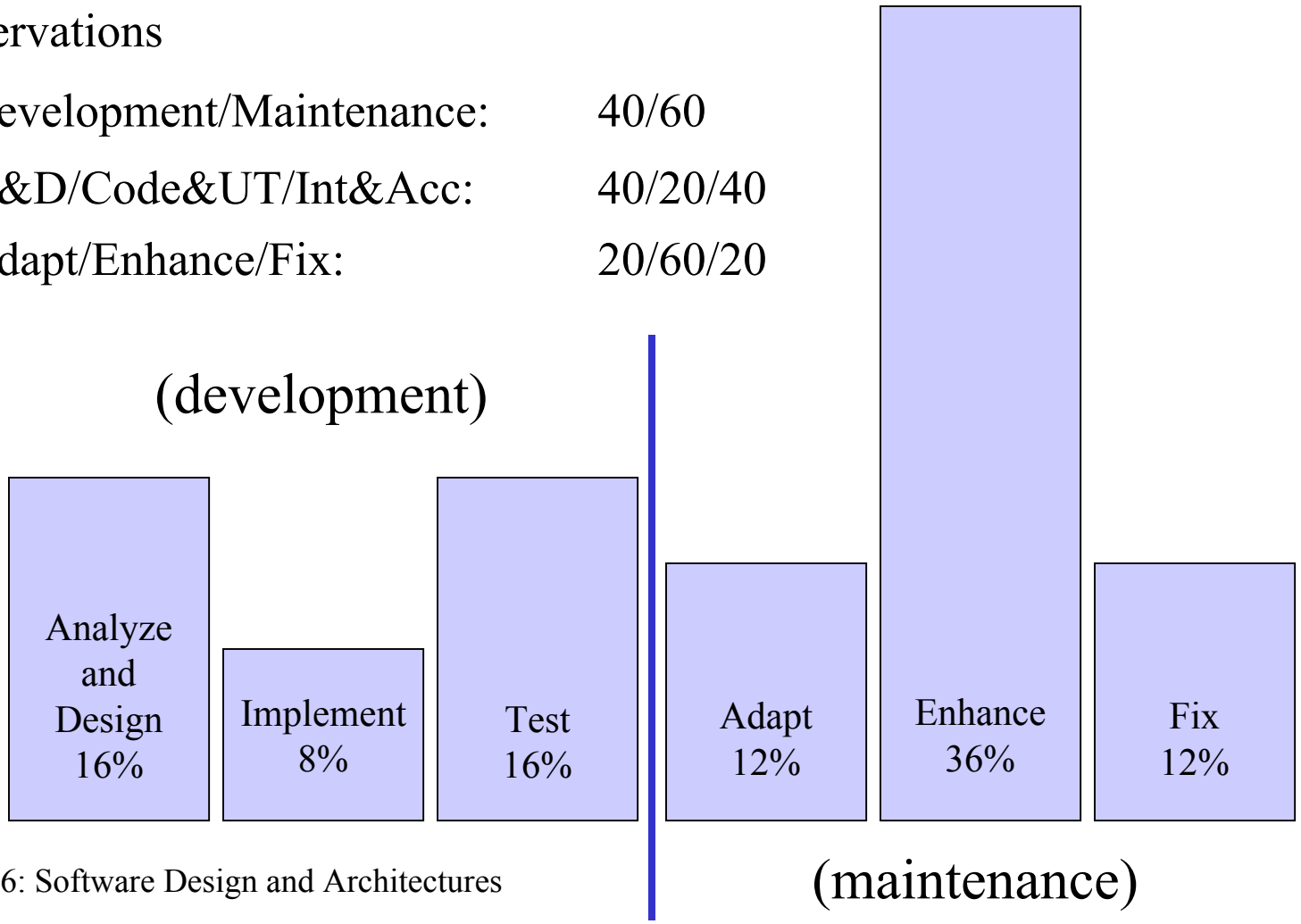
Contents of *project legacy*

1. Project description.
2. Initial expectations.
3. Current status of the project.
4. Remaining areas of concern.
5. Activities/time log.
6. Technical lessons learned.
7. Managerial lessons learned.
8. Recommendations for future projects.

Relative effort

Observations

- Development/Maintenance: 40/60
- A&D/Code&UT/Int&Acc: 40/20/40
- Adapt/Enhance/Fix: 20/60/20



Software size categories

Category	Number of programmers	Duration	Product size (LOC)
Trivial	1	1–4 weeks	500
Small	1	1–6 months	1K–2K
Medium	2–5	1–2 years	5K–50K
Large	5–20	2–3 years	50K–100K
Very large	100–1000	4–5 years	1M
Extremely large	2000–5000	5–10 years	>1M

How programmers spend their time†

- Writing programs: 13%
- Reading programs and manuals: 16%
- Job communication: 32%
- Personal: 13%
- Miscellaneous: 15%
- Training: 6%
- Mail: 5%

†Bell Labs Study (1964, 70 programmers)

Quality and productivity factors

- Individual ability.
- Team communication.
- Product complexity.
- Product size.
- Choice of languages and notations.
- Systematic approaches.
- Stability of requirements.
- Level of reliability.
- Problem understanding.
- Required skills.
- Required training.
- Management skills.
- Facilities and resources.
- Available time.

Some quality attributes

- Portability.
- Reliability.
- Correctness.
- Robustness.
- Efficiency.
- Testability.
- Understandability.
- Modifiability.
- Accuracy.
- Ease of use.
- Accountability.
- Completeness.