STANDARDS: ESA PSS-05

Introduction to the ADD Phase

Feliz Gouveia

Software Engineering

The ADD Phase

- The UR phase can be called the ‘solution phase' of the life cycle.
- The purpose of this phase is to define a collection of software components and their interfaces to establish a framework for developing the software.
- This is the ‘Architectural Design’, and it must cover all the requirements in the SRD.

ADD activities

- The principal activity of the AD phase is to develop the architectural design of the software and document it in the ADD. This involves:
  - constructing the physical model;
  - specifying the architectural design;
  - selecting a programming language;
  - reviewing the design.

Construction of the physical model

- The physical model **should** be derived from the logical model, described in the SRD.
- In transforming a logical model to a physical model, “design decisions” are made in which functions are allocated to components and their inputs and outputs defined.
- Design decisions **should** also satisfy non-functional requirements, design quality criteria and implementation technology considerations.
- Design decisions **should** be recorded.
- CASE tools **should** be used.

Decomposition of the software into components

- The software **should** be decomposed into a hierarchy of components according to a partitioning method. Examples of partitioning methods are “functional decomposition” and “correspondence with real world objects”.
- Top-down decomposition is vital for controlling complexity because it enforces “information hiding” by demanding that lower-level components behave as “black boxes”.
- Only the function and interfaces of a lower-level component are required for the higher-level design.

Implementation of non-functional requirements

- The design of each component **should** be reviewed against each of the SRD requirements. While some non-functional requirements may apply to all components in the system, other non-functional requirements may affect the design of only a few components.
- Some SRD requirements:
  - Performance requirements
  - Interface requirements
  - Operational requirements
  - Resource requirements
  - Verification requirements
  - Acceptance testing requirements
  - Documentation requirements
  - Security requirements

Design quality criteria

- Designs **should** be adaptable, efficient and understandable.
- Adaptable designs are easy to modify and maintain.
- Efficient designs make minimal use of available resources.
- Designs must be understandable if they are to be built, operated and maintained effectively.
- Designs **should** be ‘modular’, with minimal coupling between components and maximum cohesion within each component. There is minimal duplication between components in a modular design.
- Components of a modular design are often described as “black boxes” because they hide internal information from other components.

Trade-off between alternative designs

- There is no unique design for any software system.
- Studies of the different options may be necessary. A number of criteria will be needed to choose the best option. The criteria depend on the type of system.
- Only the selected design approach **shall** be reflected in the ADD (and DDD).

Specification of the architectural design

- The architectural design is the fully documented physical model.
- This **should** contain diagrams showing, at each level of the architectural design, the data flow and control flow between the components.
- Block diagrams, showing entities such as tasks and files, may also be used to describe the design.
- The diagramming techniques used **should** be documented or referenced.

Functional definition of the components

- The process of architectural design results in a set of components having defined functions and interfaces.
- The functions of each component will be derived from the SRD.
- The level of detail in the ADD will show which functional requirements are to be met by each component, but not necessarily how to meet them: this will only be known when the detailed design is complete.
- For each component the following information **shall** be defined in the ADD:
  - data input;
  - functions to be performed;
  - data output.

Definition of the data structures

- Data structures that interface components **shall** be defined in the ADD.
- Data structure definitions **shall** include the:
  · description of each element (e.g. name, type, dimension);
  · relationships between the elements (i.e. the structure);
  · range of possible values of each element;
  · initial values of each element.

Definition of the control flow

- The definition of the control flow between components is essential for the understanding of the software's operation.
- The control flow between the components **shall** be defined in the ADD.

Definition of the computer resource utilisation

- The computer resources (e.g. CPU speed, memory, storage, system software) needed in the development environment and the operational environment **shall** be estimated in the AD phase and defined in the ADD

Selection of programming languages

- Programming languages **should** be selected that support top down decomposition, structured programming and concurrent production and documentation.
- The programming language and the AD method **should** be compatible.
- Non-functional requirements may influence the choice of programming language.

Reviews

- The architectural design **should** be reviewed and agreed layer by layer as it is developed during the AD phase.
- The design of any level invariably affects upper layers: a number of review cycles may be necessary before the design of a level can be finalised.
- Walkthroughs **should** be used to ensure that the architectural design is understood by all those concerned.

OUTPUTS FROM THE PHASE

- The main outputs of the phase are the ADD and the plans for the DD phase.

The ADD

- The Architectural Design Document (ADD) is the key document that summarises the solution.
- It is the kernel from which the detailed design grows. The ADD **shall** define the major components of the software and the interfaces between them.
- The ADD **shall** define or reference all external interfaces.
- The ADD **shall** be an output from the AD phase.
- The ADD **shall** be complete, covering all the software requirements described in the SRD. To demonstrate this, a table cross-referencing software requirements to parts of the architectural design **shall** be placed in the ADD.

The ADD (1)

- The ADD **shall** be consistent. Software engineering methods and tools can help achieve consistency, and their output may be included in the ADD.
- The ADD **shall** be sufficiently detailed to allow the project leader to draw up a detailed implementation plan and to control the overall project during the remaining development phases.
- The ADD **should** be detailed enough to enable the cost of the remaining development to be estimated to within10%.

The ADD (2)

- The estimate of the total project cost (accurate to 10%), and the management plan for the DD phase, must be documented in the DD phase section of the Software Project Management Plan.
- The configuration management procedures for the documents, deliverable code, CASE tool products and prototype software, to be produced in the DD phase, must be documented in the Software Configuration Management Plan.