XV. The Requirements Specification Document (RSD)

What to include/not include in a RSD?
Attributes of a Well-Written RSD
An Example

The Requirements Specification Document (RSD)

- Produced by the requirements engineering process; describes all requirements for the system under design and is intended for several purposes:
- **Communication** among customers, users and designers;
- **Support** for system testing, verification and validation activities;
- **Control** of system evolution -- maintenance, extensions and enhancements to system should be consistent with requirements.

Contents of a RSD

- What to include in a RSD:
  - ✓A complete yet concise description of the entire external interface of the system with its environment;
  - ✓**Functional** (or **behavioural**) **requirements** specify what the system does by relating inputs to outputs;
  - ✓**Non-Functional** (**quality**) **requirements** prescribe global attributes of the system.
- What **not** to include in a RSD:
  - ✓**Project requirements** -- they are development-specific, and irrelevant as soon as the project is over.
  - ✓**Designs** -- design is irrelevant to customers.
  - ✓**Quality assurance plans** -- e.g., plans for configuration management, verification and validation, testing, etc.

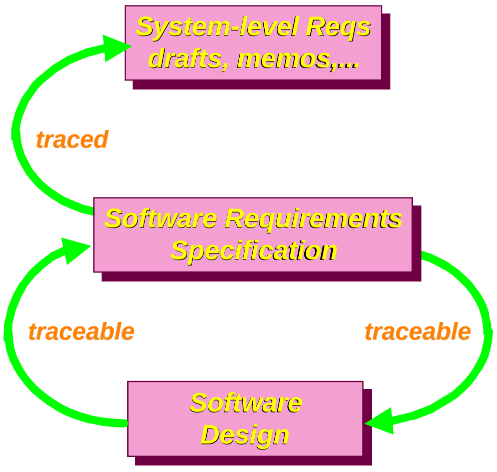
Content Qualities

- **Correct** in that all stated requirements represent a need a stakeholder has (customer, user, analyst or designer,…)
- **Unambiguous** in that every stated requirement has a unique interpretation.
- **Complete** in that it possesses the following four qualities:
  - ✓Describes everything the software is supposed to do;
  - ✓The response to input combinations is stated explicitly;
  - ✓Pages and figures are numbered;
  - ✓There are no "to-be-determined" sections.
- **Verifiable** in that every requirement can be established through a finite-cost, effective process.
- **Consistent** in that it avoids (i) conflicting behaviour, (ii) conflicting terms, (iii) conflicting attributes (iv) temporal inconsistencies

Qualities of a Well-Written RSD

- **Understandable by customers**, so formal notations can only be used as backup, while the RSD document itself is expressed in natural language or perhaps UML.
- **Modifiable** in that it can be easily changed without affecting completeness, consistency; a table of contents (TOC) helps, so does an index and cross references where appropriate.
- **Traced** in that the origin of every requirement is clear; this can be achieved by referencing earlier documents.
- **Traceable** in that attributes of the design can be traced back to requirements and vice versa; to enhance traceability (i) number every requirement, (ii) number every part of the RSD hierarchically, all the way down to paragraphs

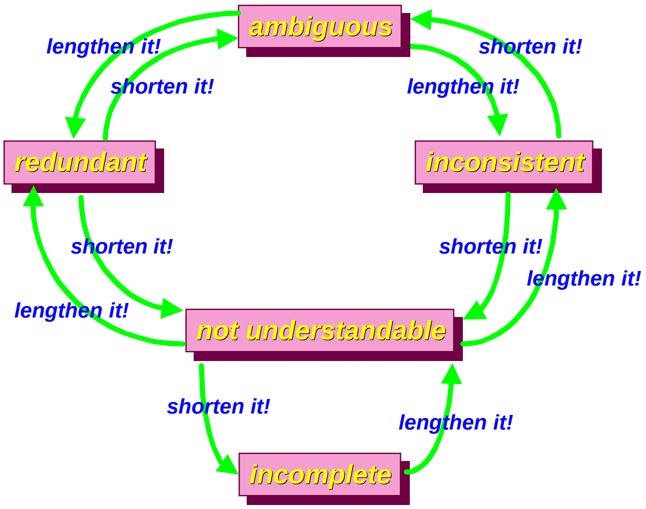
Traceable vs Traced



Style Qualities

- **Design-independent** in the sense that it does not imply a particular software architecture or algorithm
- **Annotated** in that it provides guidance to the developers; two useful types of annotations are (i) relative necessity, i.e., how necessary is a particular requirement from a stakeholder perspective, (ii) relative stability, i.e., how likely is it that a requirement will change.
- **Concise** -- the shorter the better!
- **Organized** in the sense that it is easy to locate any one requirement.

There is no Perfect RSD!



ambiguous

lengthen it!

shorten it!

shorten it!

lengthen it!

redundant

inconsistent

shorten it!

lengthen it!

shorten it!

lengthen it!

not understandable

shorten it!

lengthen it!

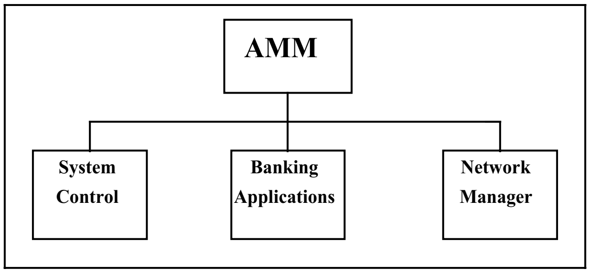
incomplete

How to Organize a RSD

- There are many RSD standards, including: US DoD DI-MCCR-80025A, IEEE ANSI 830-1984, etc.
- Organization may be based on different criteria:
  - ✓External stimulus or external situation, e.g., for an aircraft landing system, wind gusts, no fuel,...;
  - ✓System feature, e.g., call forward,...;
  - ✓System response, e.g., generate pay-cheques;
  - ✓External object, e.g., by book type for a library;
  - ✓User type/use case.
- It is useful to define a hierarchy among these criteria, use it throughout the RSD document, e.g., sections are defined with respect to (wrt) external stimulus, subsections wrt system feature etc.

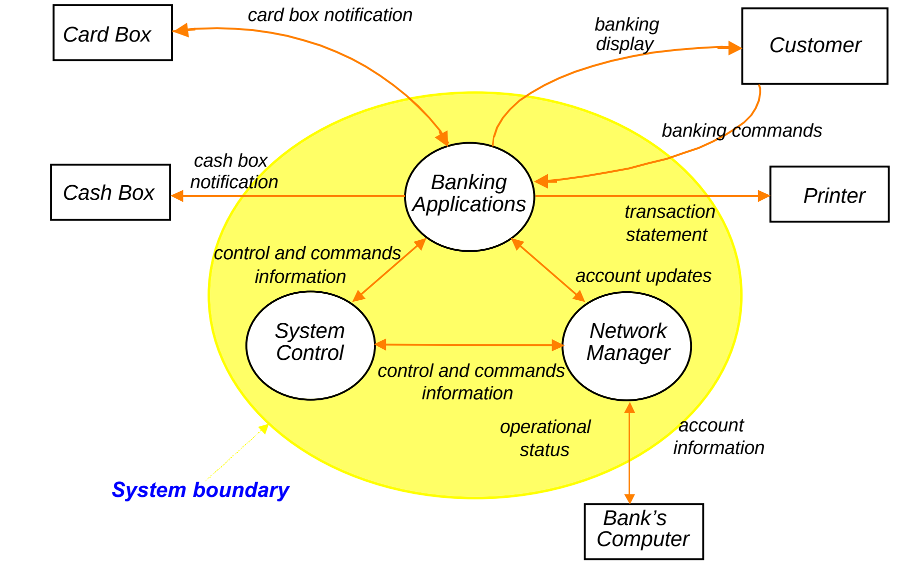
Example System Decomposition

An Automated Money Machine (AMM) might be decomposed as follows:



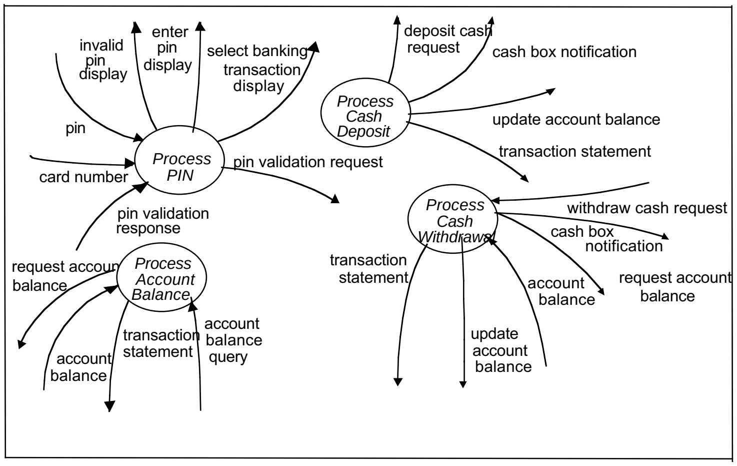
Banking Applications handles banking transactions.
Network Manager communicates with central system.
System Control is responsible for startup/shutdown control of the AMM system and error handling.

Interfaces, I/O



Software Requirements

Define I/O for each use case for the Banking Applications



References

- [Davis93] Davis, A., Software Requirements, Prentice-Hall, 1993, (chapter 3)
- [Thayer90] Dorfman, M. and Thayer, R. Standards, Guidelines and Examples on System and Software Requirements Engineering, IEEE Computer Society Press, 1990.