IF2261 Software Engineering

System Engineering

Program Studi Teknik Informatika
STEI ITB

Overview

- Before software can be engineered:
  - the system it is part of must be understood,
  - the overall objective of the system must be determined,
  - the role of the system elements (hardware, software, people, data, etc.) must be identified, and
  - the operational requirements must be created.
- Don't take a "software-centric" view of the system; consider all system elements before focusing on software.
- Good system engineering begins with a clear understanding of the "world view" and progressively narrows until technical detail is understood.
- Complex systems are actually a hierarchy of macro-elements that are themselves systems.

** SEPA 6th ed, Roger S. Pressman*

System – Definition

Webster's Dictionary

- *A set or arrangement of things so related as to form a unity or organic whole*
- *A set of facts, principles, rules, etc., classified and arranged in an orderly form so as to show a logical plan linking the various parts*
- *A method or plan of classification or arrangement*
- *An established way of doing something; method; procedure….*
- …..
- ….

Computer-Based Systems

[PRE2005]

- *A set or arrangement of elements that are organized to accomplish some predefined goal by processing information*
- The goal:

  To support some business function or to develop a product that can be sold to generate business revenue
- To accomplish the goal, a computer-based system makes use of a variety of system elements

Computer-Based System Elements

- Software
- Hardware
- People
- Database
- Documentation
- Procedures

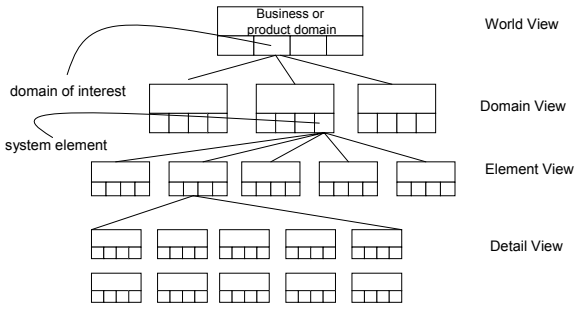
** SEPA 6th ed, Roger S. Pressman*

System Engineering Hierarchy

- World view → WV = $\{D_1, D_2, D_3, \ldots, D_n\}$
  - Composed of a set of domains (D_i) which can be each be a system or system of systems
- Domain view → DV = $\{E_1, E_2, E_3, \ldots, E_m\}$
  - Composed of specific elements (E_i) each of which serves some role in accomplishing the objective and goals fo the domain or component
- Element view → EV = $\{C_1, C_2, C_3, \ldots, C_k\}$
  - Each element is implemented by specifying the technical component (C_k) that achieve the necessary function for an element
- Detail view

** SEPA 6th ed, Roger S. Pressman*

System Engineering Hierarchy



System Modeling

The engineer creates models that:

- Define the processes that serve the needs of the view under consideration
- Represent the process behavior and the assumptions on which the behavior is modeled
- Explicitly define the exogenous (links between constituents) and endogenous (links between constituent components) input to the model
- Represent all linkages (including outputs) required to understand the view

** SEPA 6th ed, Roger S. Pressman*

System Model Restraining Factors

- Assumptions
  - reduces the number of possible variations
- Simplifications
  - enable the model to be created in a timely manner
- Limitations
  - help to bound the system
- Constraints
  - guide the manner in which the model is created and implemented
- Preferences
  - indicate the preferred architecture for all data, functions, and technology

** SEPA 6th ed, Roger S. Pressman*

Systems Modeling Process

Hartley-Pirbhai Modeling

- System Context Diagram (SCD)
  - top level node in system hierarchy used to establish the boundary between the system being implemented (system model template serves as its basis)
- System Flow Diagram (SFD)
  - refinement of the process and control functions from SCD, derived by identifying the major subsystems and lines of information flow
- Initial SFD is becomes the top level node of a hierarchy of more successively more detailed SFD's
- System Specification
  - developed by writing narrative description for each subsystem and definitions for all data that flow between subsystems

** SEPA 6th ed, Roger S. Pressman*

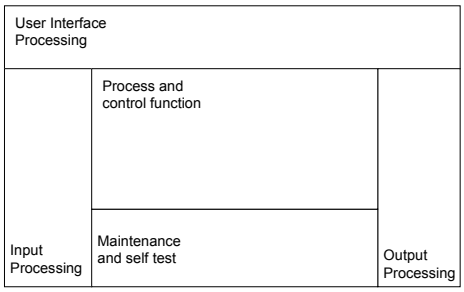
System Model Template

Hartley-Pirbhai Modeling

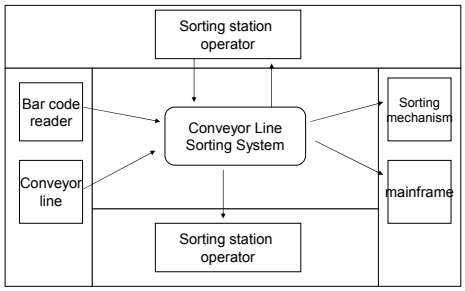
- User interface
- Input
- Process and control functions
- Output
- Maintenance and self test

** SEPA 6th ed, Roger S. Pressman*

The Template



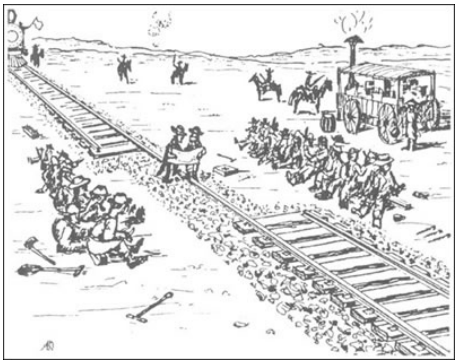
SFD for CLSS



System Modeling with UML

- Deployment diagram
  - depicts hardware elements that are part of the physical architecture of the system
- Activity diagram
  - used to represent the procedural aspects of the system software elements, similar to a flowchart in that system functions are shown as nodes, decision points are shown as diamonds, and arrows are used to show flow through the system
- Class diagram
  - shows the class attributes and operations that may be applied to the class within the context of the system
- Use-case diagram
  - illustrates the manner in which an actor interacts with the system, each labeled oval within a system boundary represents one text scenario or use-case

** SEPA 6th ed, Roger S. Pressman*



Business Process Engineering

- Goal
  - to define architectures that will enable a business to use information effectively
- Architectures
  - Data architecture
    - provides framework for information needs of a business or business function
  - Applications architecture
    - those system elements that transform objects within the data architecture for some business purpose
  - Technology infrastructure
    - provides foundation for the data and application architectures

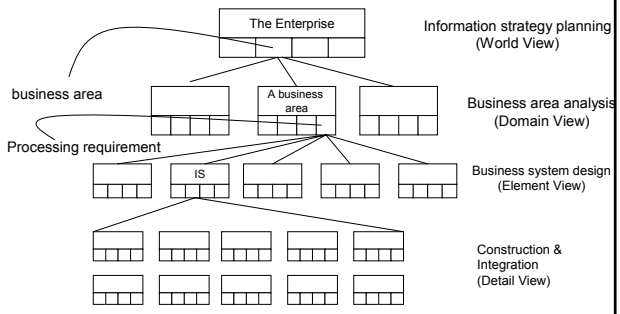
** SEPA 6th ed, Roger S. Pressman*

Business Process Engineering (2)

- Hirarchy
  - Information Strategy Planning (world view)
  - Business Area Analysis (domain view)
  - Business System Design (element view - software engineers)
  - Construction and Integration (detailed view - software engineers)

** SEPA 6th ed, Roger S. Pressman*

The Business Process Engineering Hierarchy

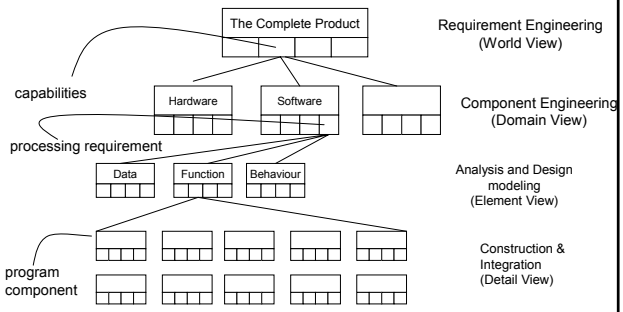


Product Engineering

- Goal
  - to translate the customer's desire for a set of defined capabilities into a working product
- Hirarchy
  - Requirements engineering (world view)
  - Component engineering (domain view)
  - Analysis and Design modeling (element view - software engineers)
  - Construction and Integration (detailed view - software engineers)

** SEPA 6th ed, Roger S. Pressman*

The Product Engineering Hierarchy

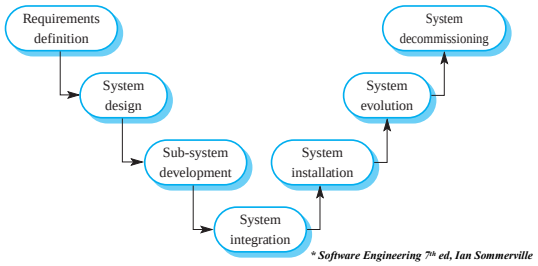


Systems engineering

- Specifying, designing, implementing, validating, deploying and maintaining socio-technical systems.
  - Systems that include technical systems but also operational processes and people who use and interact with the technical system. Socio-technical systems are governed by organisational policies and rules.
- Concerned with the services provided by the system, constraints on its construction and operation and the ways in which it is used.

** Software Engineering 7th ed, Ian Sommerville*

The Systems Engineering Process



** Software Engineering 7th ed, Ian Sommerville*

System Requirements Definition

- Three types of requirement defined at this stage
  - Abstract functional requirements.
    - System functions are defined in an abstract way;
  - System properties.
    - Non-functional requirements for the system in general are defined;
  - Undesirable characteristics.
    - Unacceptable system behaviour is specified.
- Should also define overall organisational objectives for the system.

** Software Engineering 7th ed, Ian Sommerville*

System Requirements Problems

- Complex systems are usually developed to address wicked problems
  - Problems that are not fully understood;
  - Changing as the system is being specified.
- Must anticipate hardware/communications developments over the lifetime of the system.
- Hard to define non-functional requirements (particularly) without knowing the component structure of the system.

** Software Engineering 7th ed, Ian Sommerville*

The System Design Process

- Partition requirements
  - Organise requirements into related groups.
- Identify sub-systems
  - Identify a set of sub-systems which collectively can meet the system requirements.
- Assign requirements to sub-systems
  - Causes particular problems when COTS are integrated.
- Specify sub-system functionality.
- Define sub-system interfaces
  - Critical activity for parallel sub-system development.

System Design Problems

- Requirements partitioning to hardware, software and human components may involve a lot of negotiation.
- Difficult design problems are often assumed to be readily solved using software.
- Hardware platforms may be inappropriate for software requirements so software must compensate for this.

Requirements and Design

- Requirements engineering and system design are inextricably linked.
- Constraints posed by the system's environment and other systems limit design choices so the actual design to be used may be a requirement.
- Initial design may be necessary to structure the requirements.
- As you do design, you learn more about the requirements.

Sub-System Development

- Typically parallel projects developing the hardware, software and communications.
- May involve some COTS (Commercial Off-the-Shelf) systems procurement.
- Lack of communication across implementation teams
- Bureaucratic and slow mechanism for proposing system changes means that the development schedule may be extended because of the need for rework.

System Integration

- The process of putting hardware, software and people together to make a system.
- Should be tackled incrementally so that sub-systems are integrated one at a time.
- Interface problems between sub-systems are usually found at this stage.
- May be problems with uncoordinated deliveries of system components.

System Installation

- After completion, the system has to be installed in the customer's environment
  - Environmental assumptions may be incorrect;
  - May be human resistance to the introduction of a new system;
  - System may have to coexist with alternative systems for some time;
  - May be physical installation problems (e.g. cabling problems);
  - Operator training has to be identified.

System Evolution

- Large systems have a long lifetime. They must evolve to meet changing requirements.
- Evolution is inherently costly
  - Changes must be analysed from a technical and business perspective;
  - Sub-systems interact so unanticipated problems can arise;
  - There is rarely a rationale for original design decisions;
  - System structure is corrupted as changes are made to it.
- Existing systems which must be maintained are sometimes called legacy systems.

** Software Engineering 7th ed, Ian Sommerville*

System Decommissioning

- Taking the system out of service after its useful lifetime.
- May require removal of materials (e.g. dangerous chemicals) which pollute the environment
  - Should be planned for in the system design by encapsulation.
- May require data to be restructured and converted to be used in some other system.

** Software Engineering 7th ed, Ian Sommerville*