Integrative Systems Engineering

Requirements Engineering in the Solution Domain

Prof. Michael Zink
Overview

- What is the solutions Domain?
- Producing the system model
- Internal functionality
- Human interaction functionality
- System transactions
- Modes of operations
- Banking example
- Car example
- From requirements to subsystem
What is the Solution Domain?

- Domain in which engineers use their ingenuity to solve problems

- It must be allowed for creativity and ingenuity to work together
What is the Solution Domain?

- First, transform stakeholder requirements into set of systems requirements
  - Define what system must do in order to solve problems posed by stakeholder requirements
- System model must be created at level of abstraction that
  - Enables functionality of the system to be defined
  - Without going into unnecessary detail
- Architectural design
- Problem:
  - Too much detail added on to soon!
What is the Solution Domain?

- **Stakeholder Requirements**
  - Define System Requirements
  - System Requirements
  - Create Design Architecture
  - System Components Requirements (Subsystem Requirements)

- **Engineer Requirements**
  - Create Subsystem Architecture
  - System Components Requirements (Subsystem Requirements)

- **Acceptance Strategy**
  - System Model
  - Analysis Results
  - System Test Strategy
  - Subsystem Test Strategy
  - Subsystem Component Test Strategy

- **System Architecture Model**
  - Analysis Results
From Stakeholder Reqs. to Systems Reqs.

- Stakeholder Requirements
- Agree Requirements
- Qualification Strategy
- System Requirements
- Agree Requirements
- Qualification Strategy

- Derive Requirement & Qualification Strategy
- Analyze & Model
- Change Request
- Analysis Results
- System Models
- Change Request
- Change Request
Producing the System Model

- Work in context of model to avoid tendency to go into much detail
- Level of detail should be adequate to development level at which requirements engineering is been done
- Engineers are not comfortable with working on abstract system model
- Components should be specified as "black boxes" – internal details are of no concern
Creating System Models to Derive Systems Req.

- System model must be created at appropriate level of abstraction such that it encompasses:
  - Internal functionality system must exhibit – *what* system must do rather than *how* it should be done
  - Functionality necessary to enable the system to interact with other systems in its environment
  - Functionality necessary to enable people to successfully interact with it
  - Functionality to prevent system from malfunctioning owing to the presence of other systems (threats) in its environment – must also prevent system from interfering in an adverse way with environment
Creating System Models to Derive Systems Req.

- Context of system within its environment must be defined with respect to:
  - Existing system which with new system is required to cooperate
  - Types of people who are intended to interact with system
  - Threats system must defend against
  - Adverse effects that must be prevented

![Diagram of system models and requirements](image-url)
Creating System Models to Derive Systems Req.

- In Practice use several models to cover aspects
- Information for different models might be the same, if it changes for one, changes have to be also made for other models
- One way to indicate models that share information is the usage of Venn diagrams

![Venn Diagram with models A, B, C, D, E and System Information Domain]
Internal Functionality

- Necessary to create structure or model as basis for creating system requirements
- Form of decomposition of the system into modules or high-level components, such as subsystems
- Often term “Object” is used to define decomposition element
- Use of object-oriented approach can also make creation of traceability from system requirements to stakeholder requirements easier task
Internal Functionality

- System model may also be required to indicate intended behavior of system
- Message sequence charts and behavior diagrams are often used to illustrate behavior
- Information handling
- E.g. radar system:
  - Flight plan has long lifetime
  - Current position of an aircraft is soon out of date
- Examination:
  - Longevity of information – how long relevant?
  - Freshness of information – how up to date?
Interface Functionality

- Necessary to define interactions required with other systems
- Interactions may involve movement of information or material between systems
  - Uni- or bi-directional
  - Limits on capability
  - Time response requirement
- Nature of interfaces varies significantly
  - Baseline reference that indicates what each party undertakes to do or provide as part of the interface
  - Obligations are often documented in Interface Control Document (ICD)
- Interface control authority must be clearly defined
Human Interaction Functionality

- What interactions are going to be required?
- Context in which user will work is important
- Example:
  - User in office or lab will be warm and able to work without gloves
  - Users in harsh environments (cold weather or hazardous situation) where protective clothing will be necessary
- Design of displays and keyboards must take situation into account
Safeguard Functionality

- Environment in which system will operate in
  - In banking system it must be assured that information and money is not given to unauthorized people
  - Have car stop when brake pedal is operated

- Competing systems
  - Other systems operating in environment that may be competing with system being developed
  - Worry that cell phone communication on board a plane could interfere with navigation system
System Transactions

- Figure below indicates that there can be other transactions with external systems
- System transactions encourage Systems Engineers to stand back and take a “holistic” view of system
Modes of Operation

- Different functionality may be required in some circumstances
  - E.g., how can an air traffic controller be trained without compromising the actual system
  - System has end users with very different needs: Harbor for freight and passenger ships
  - Fallback modes of operation following a failure
  - Related to use scenarios in the stakeholder requirements
Additional Constraints

- Safety and certifiability
  - Can introduce additional requirements
- Relevant authorities have to be convinced that system is safe:
  - For use or deployment
  - E.g., certified for airworthiness in an aircraft or satellite
- Constraints by need to manufacture a system
  - Use external facility
  - Design may have to be changed to reduce cost of manufacturing
Banking Example

- Example of management information system
- Primary concern will be to model information that must be handled (while it is clear that many other areas should be addressed)
- Several system models are likely to be used:
  - Focusing on information
  - Focusing on flow
  - Focusing on security of information
Internal Functionality

- Supporting services as: current accounts, deposit accounts, loans and investment portfolios
- System must be able to collect, update and retain information
- Of vital importance are:
  - Types of information and their relationships
  - Longevity, freshness and volume of each type
- Important to determine how information is collected, disseminated and manipulated
- Number of sources of information and/or transactions
- Overall load of the system
Interface Functionality

- Primary interfaces of the system:
  - Other banks for funds transactions and use of their teller machines
  - Existing system for clearing cheques that is jointly owned between banks
  - Telecommunication service from external provider
Human Interaction Functionality

- Following list covers variety of user types for banking system:
  - *General public* – uses teller machines and online facilities
  - *Counter staff* – quick access to system in order to provide fast and efficient service
  - *Managers at various levels* – access to high level information (summary type of information by looking at a wider set of information)
  - *Policy makers and marketing staff* – E.g., capability to start new business products
  - *System maintainers* – must be able to maintain system (ideally while it is running)
Safeguard Functionality

- Security is of paramount importance
- Key element is needed to protect integrity of information that is heart of the business
  - PIN on credit and debit cards
  - Encryption for transfers between branches
- Need to keep system working in case of computer faults, power failures or communication failure
- Threads from hackers or embezzlers
  - Categories are related to perception of risk
  - Degree that can be afforded to mitigate risk depends critically on the perceived exposure
System Transactions

- Each type of user likely to be a stakeholder
- Likely there will be a set of use scenarios for each type of user
  - For the bank customer: withdrawals, deposits, and transfers (made in person or via the web or ATM)
  - Less frequently used transactions as negotiation on personal loan or mortgage
- Consider load for each type of user to estimate response time
  - Will not be a fixed time but will depend on current loading
Car Example

- Example addresses a more physical type of system
- Interesting to see that same categories of information are still present
- Issue: how to make physical model abstract?
  - Unlikely that new car will be radically different from existing ones
  - May well make reference directly to physical objects of architecture
- However, where aspects of new car are likely to be rather different (e.g., electronic vehicle control system) remaining more abstract will present advantages
Key questions that should have been addressed in stakeholder requirements:

- How many people?
- How much luggage?
- How comfortable will the car be?
- How fast will the car travel?
- How quickly will the car accelerate?
- How much will it cost?
- What information will be provided to the driver?
- What in-car entertainment facilities will be provided?
- What safety features will be necessary?
- Where will the car be used?
Internal Functionality

- Key requirements at the functionality level:
  - *Acceleration rate of the car* – Balance between engine power, overall weight, wind resistance, and drag induced by wheels
  - *Range of the car* – Balance between fuel efficiency, fuel capacity, manual or automatic gear box, and way in which car is driven
  - *Comfort level of car* – Will influence cost and weight; different clients may perceive end result differently

- These key aspects are not independent

- It is necessary to determine:
  - Whether to make a selection at all at this point, or
  - Whether to keep all options open or whether to provide a customer selectable options
Interface Functionality

- One might expect that car is going to be isolated in terms of needs to interface with other systems
- This is not the case:
  - Radio receiver which entails conforming to certain standards of demodulation
  - GPS signal decoding
  - Interface to traffic information systems
- Interface for servicing cars is also important
  - Allow service technician easy access to information for diagnosis
  - Example for a test system that is partly installed in operational system and partly in service garage
Human Interaction Functionality

- Many aspects on “user interface” for car have evolved over years
  - Relative position of food pedals – identical all over the world
  - Local conventions for other user interfaces – driver position, position and colors of indicators, etc.

- On the other hand no conventions for many other interfaces
  - Navigation system, entertainment system, etc.

- Interface must be designed for easy use
  - User guide is only “training” for user
  - Impossible to send them for training
Safeguard Functionality

- Ensure safety of car and its occupants
- Theft prevention
- Safety functionality – example braking system
  - Dual-circuit hydraulic brakes
  - System model: include implementation directly, alternatively, or just simple need for breaking
  - Assumes tacitly decision that braking will be realized via hydraulics
- Other examples are ABS and airbags
- Security:
  - Doors an locks
  - Alarm system and engine immobilizer
  - Limiting factor is cost
  - Outside factors: Competitors, insurance companies
System Transactions

- Transactions for cars are based on journeys, for example:
  - Driver, shopping trip in town – leave parking bay, travel, park, secure vehicle, unlock vehicle, leave parking bay, travel, park, unload, secure vehicle
  - Driver, highway trip
  - Driver, airport trip (with luggage)
  - Driver, trip with accident
  - Passenger – get in, fit belt, travel, undo belt, get out
  - Garage technician – repeatedly service, with major/minor intervals
  - Owner – buy, depreciate, sell/dispose
  - Salesman – repeatedly attempt to sell, ended by selling, warranty period
Modes of Operation

- Prevailing terrain could influence mode in which car operates (e.g., 2- or 4-wheel drive)
  - 2- or 4-wheel drive
  - Mixture between air and gas
- Maintenance mode
  - Management system is downloading collected information for analysis
- More futuristic would be the mode in which car joins highway “train”
  - Set of cars which all travel at same speed
  - Minimal spacing
  - Automatically controlled
Create Document Structure for Requirements

- Start deriving requirements from models
- Challenge is to find structure into which all derived requirements can be placed
  - Every requirement has an obvious place
  - Empty sections are empty by design and not by accident
- Choose one of the models as primary source for documentation structure
  - Choose model with the widest scope
  - Data model in banking system
  - Physical model for car
Derive or Allocate Requirements

- Once structure has been agreed, derived requirements can be collected and placed in structure
- Acceptance criteria should be documented with each requirement
- It is also vital to consider how testing or demonstration can be organized
  - Consider special test equipment
  - May require separate development
  - Built-in-tests and monitor points
  - E.g., built-in test for electronic system in car, oil pressure gauge
Sources of constraints:

- **Design decisions** – e.g., the decision to have a dual hydraulically operated braking system
- **The application itself** – e.g., that the equipment must be able to cope with the vibration generated by the car when it is in motion
- **Safety** – e.g., how can the developer convince the authorities that the car will not constitute a hazard to other road users
- **Manufacturing** – e.g., can the car be manufactured using the existing facilities at a reasonable cost?
From System Requirements to Subsystem

- System Requirements
- Agree Requirements
- Qualification Strategy
- Change Request
- Analyze & Model
- System Architecture Model
- Analysis Results
- Change Request
- Derive Stakeholder Requirement & Qualification Strategy
- Change Request
- Agree Requirements
- Qualification Strategy

Subsystem Requirements
From System Requirements to Subsystem

- Create design architecture whose components are the major subsystems of proposed system
- Process starts off by agreeing on input requirements with customer
- Review criteria for system requirements must be used as basis for agreement process
- Requirements should be free from implementation bias
- Analysis work necessary to support the agreement process helps to educate designer about what is intended – start thinking about possible solutions
Creating a System Architecture Model

- Architecture model identifies components of system
- Designer must understand how components work together to develop emergent properties of system
- Designer may find that it is impossible to satisfy requirements at all or at reasonable cost
- After design exists, costs and development time can be predicted with greater accuracy
- Worthwhile considering two or more alternative designs
- Each component of architecture must be described in terms of internal functions and its interactions
Deriving from Architectural Design Model

- Requirements can be derived from description of components
- Requirements must address:
  - Functionality that the component must provide
  - Interfaces it must use or provide
  - Constraints to which component must conform
- Component requirements are essentially system requirements for that component
- Each requirement should be traced backed to one or several input requirements that it satisfies
- Strategy for testing each component must be determined