

Welcome

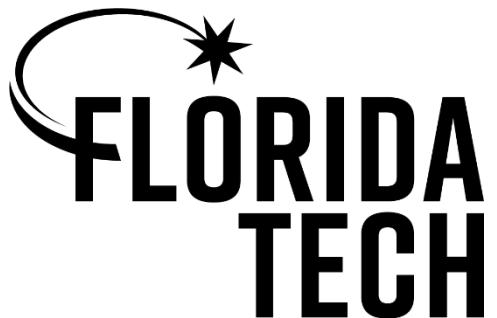
SYS 5460: System Requirements Analysis

Course Admin

- Syllabus
- Assignments
- Overview of CANVAS site
- Available literature
- Software usage
- Expectations and prior work on requirements

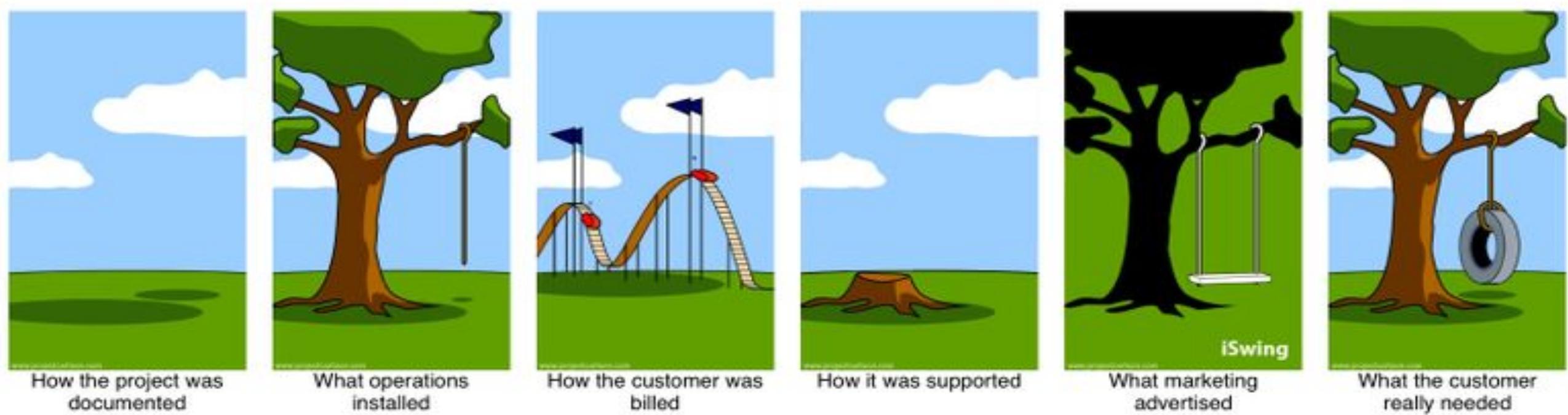
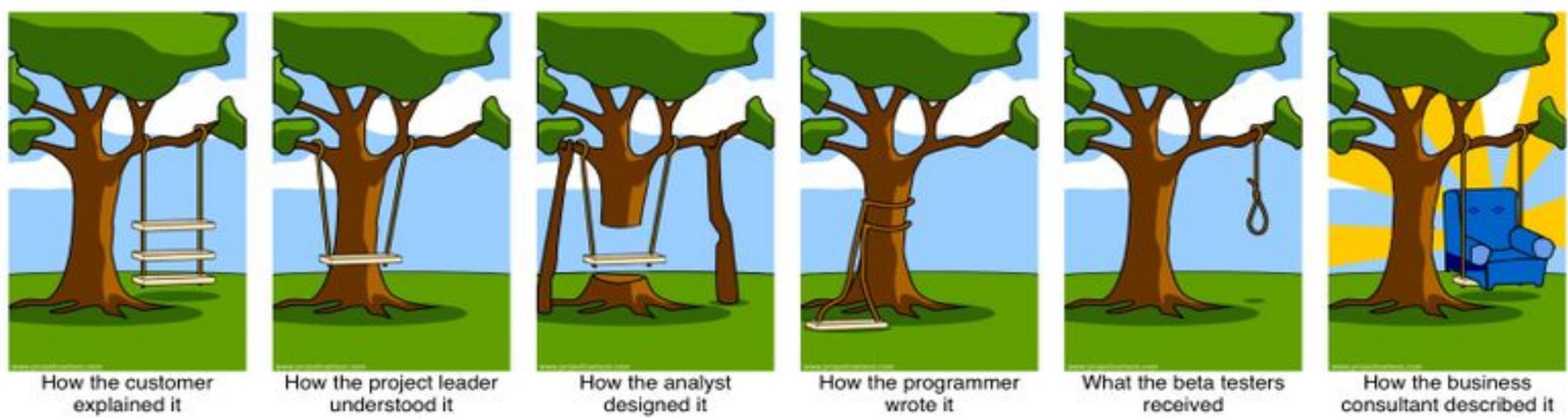
**Department of Computer &
Engineering Sciences**

College of Engineering
Florida Institute of Technology



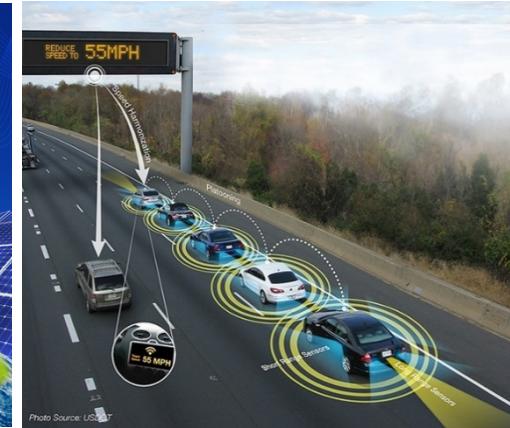
Introduction

- Requirements and Systems Engineering
- Requirement Definition
- Requirements and Project Management
- System Interfaces
- Emergent Properties of a System
- Discussion



Motivation for Systems Engineering

- High expectation for system performance
- Competitive pressure
 - Increased Capabilities
 - Reduced Costs
 - Shorter lifecycles
- Increased complexity, System of Systems
- **Systems engineering** approach used to provide solutions to complex problems



Reasons for Project Failure

| | |
|----------------------------------------|-------|
| * Incomplete requirements | 13.1% |
| * Lack of user involvement | 12.4% |
| Lack of resources | 10.6% |
| * Unrealistic expectations | 9.9% |
| Lack of executive support | 9.3% |
| * Changing requirements/specifications | 8.7% |
| Lack of planning | 8.1% |
| * Didn't need it any longer | 7.5% |

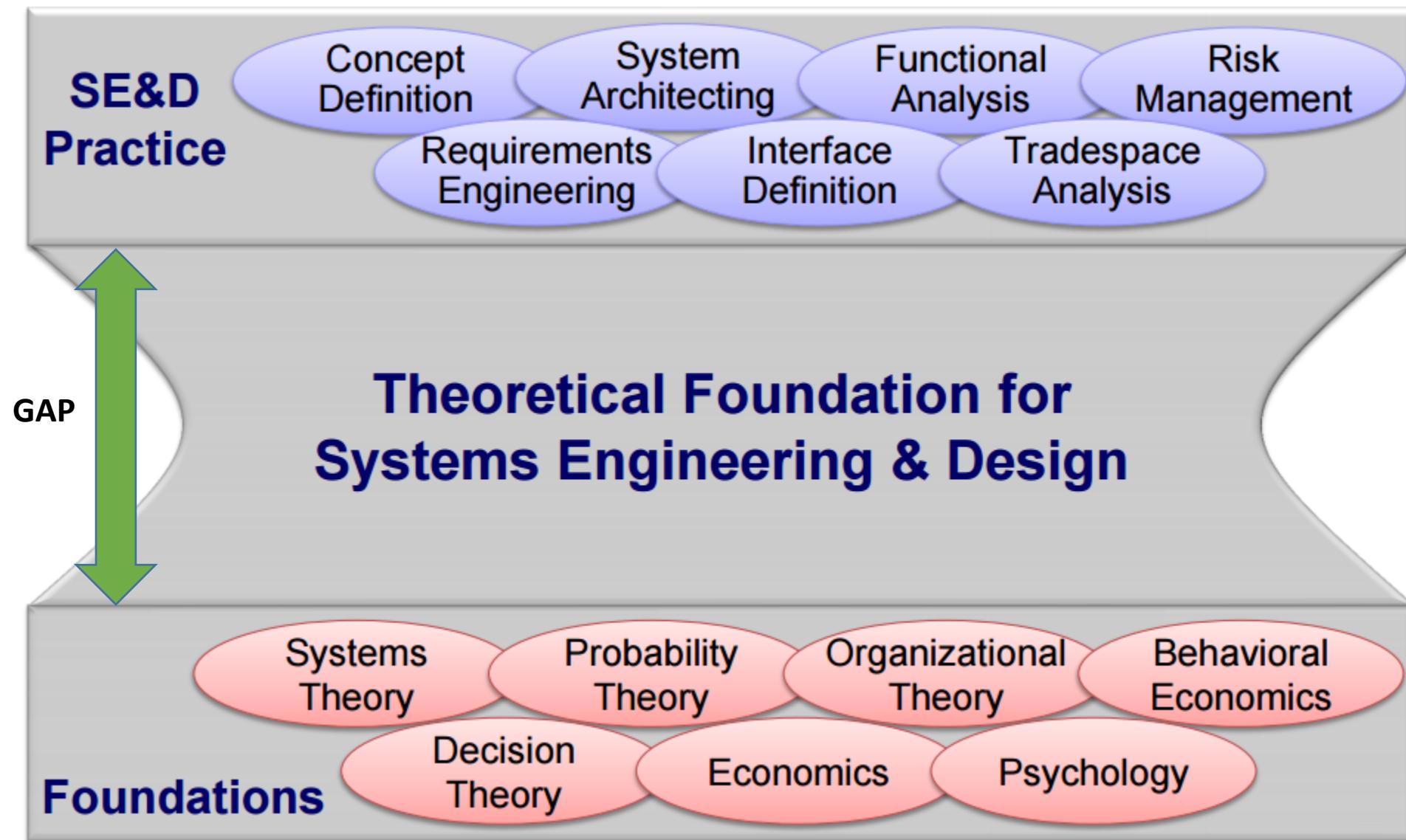
Standish Group 1995 & 1996
Scientific American, Sept. 1994

Reasons for Project Success

| | |
|-----------------------------------|-------|
| * User involvement | 15.9% |
| Management support | 13.9% |
| * Clear statement of requirements | 13.0% |
| Proper planning | 9.6% |
| * Realistic expectations | 8.2% |
| Smaller milestones | 7.7% |
| Competent staff | 7.2% |
| * Ownership | 5.3% |

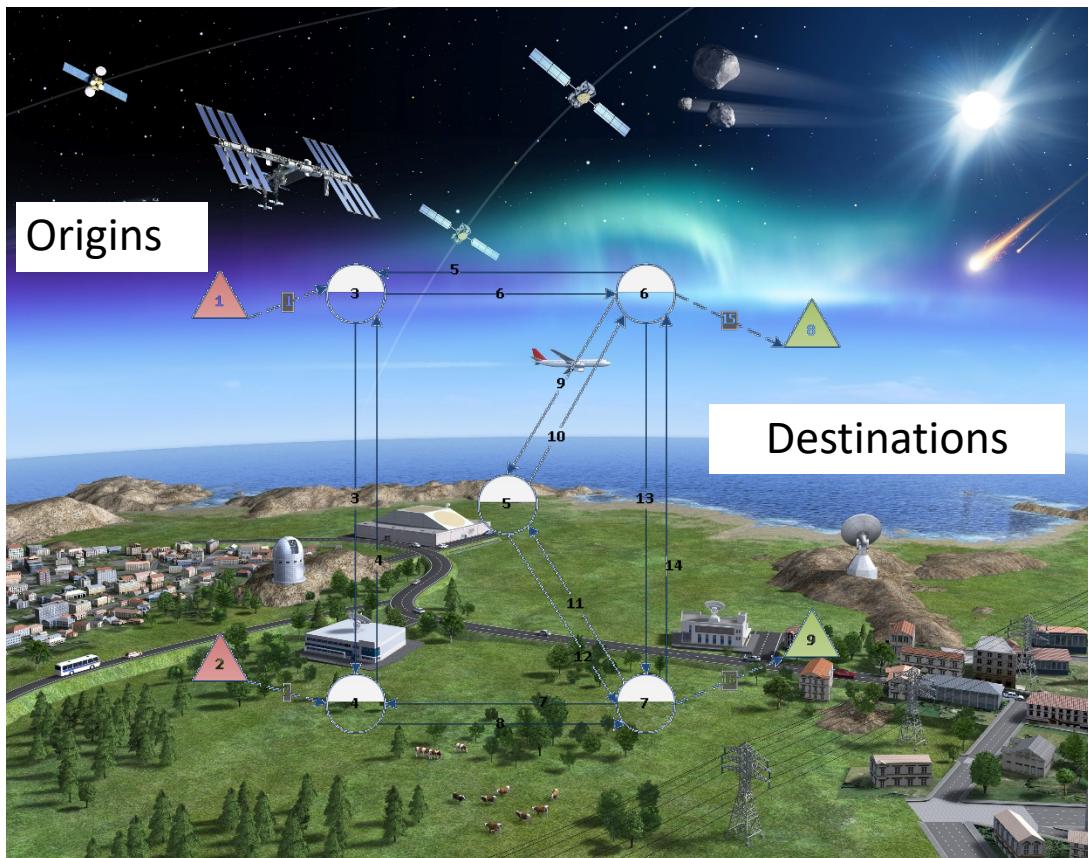
Standish Group 1995 & 1996

Scientific American, Sept. 1994



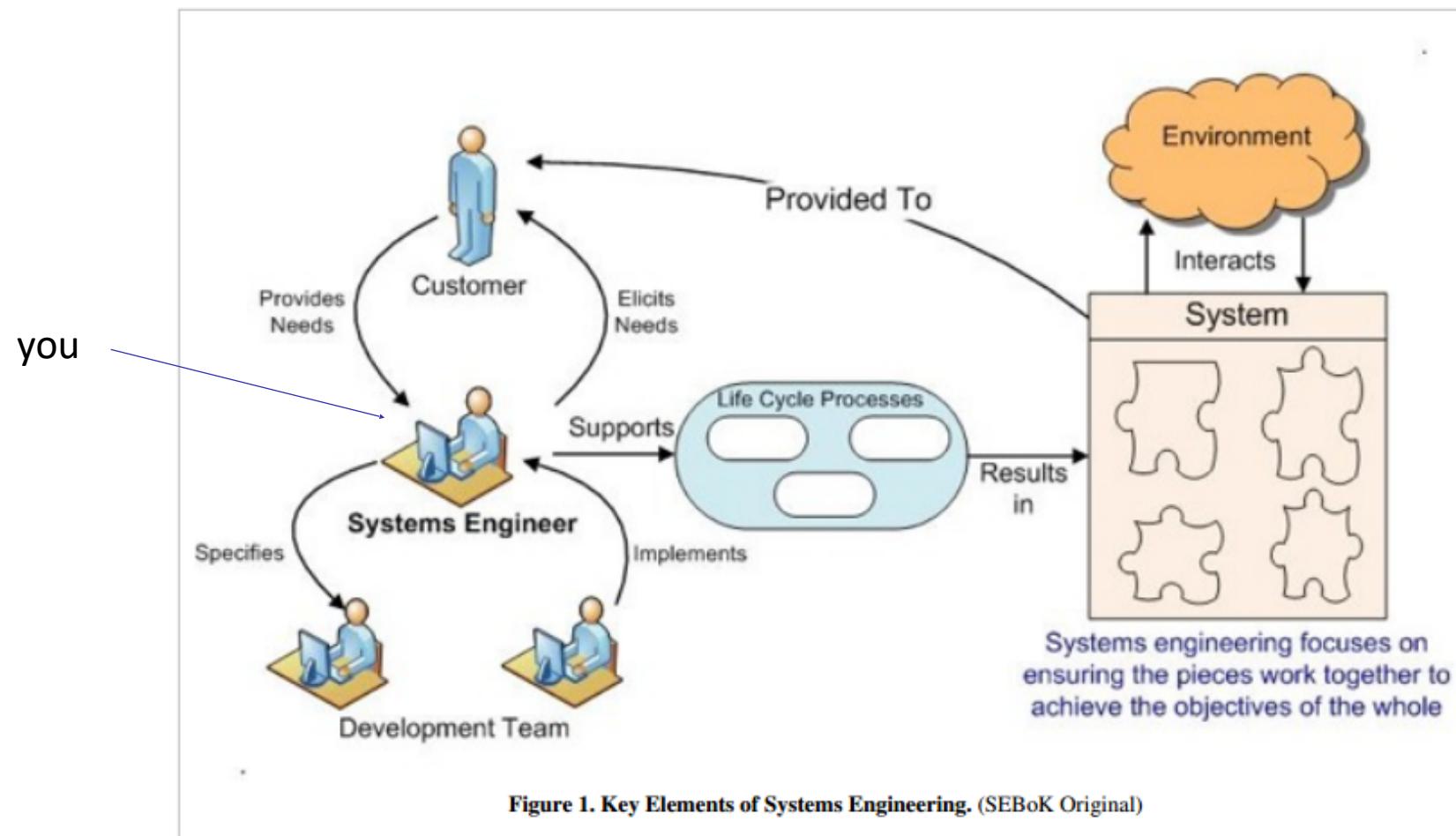
Optimization Model Examples: Optimal Routing

- Consider a communication network consisting of communication links and nodes
- Users need to send data from origin nodes to destination nodes
- A node switches arriving data onto one of the links incident to it
- Routers and protocols are used for these purpose



Systems Engineer

- A systems engineer helps ensure the elements of the system fit together to accomplish the objectives of the whole, and ultimately satisfy the needs of the customers and other stakeholders who will acquire and use the system (SEBoK)

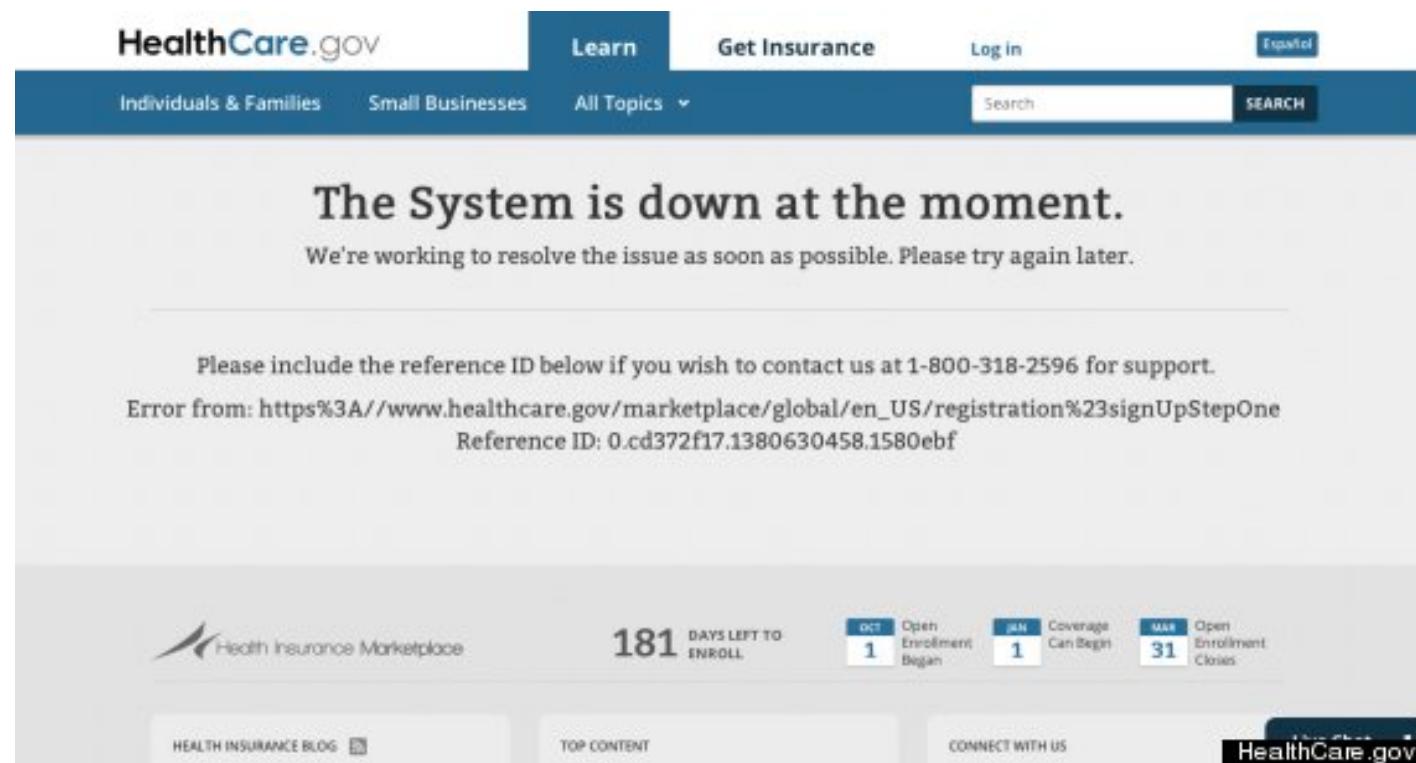


Importance of Requirements

- Requirements Engineering is common sense, but it is perceived to be difficult and is not well understood
- Systems engineering is critical in today's industry and requirements engineering is an important stage of that overall process
- In a global competitive market “**time to market**” and meeting stakeholder requirements are key success factors
- Requirements engineering is also about management. Requirements can be used to manage systems development

Time to Market

- “time to market” is not sufficient.
- The real goal is “time to market with the right product”
- Establishing the requirements enables us to agree on and visualize the “right product”.



Requirements - Importance | TTM

- Time to market must couple with the right product
- Establishing the requirements enables us to agree on and visualize the right product



- What requirement was missing?

Requirements - Importance | TTM

- Time to market must couple with the right product
- Establishing the requirements enables us to agree on and visualize the right product



Applications must be submitted electronically by 5:00 pm on Wednesday, March 6, 2013.

Please visit:
http://agency.governmentjobs.com/career/default.cfm?action=view.job&id=601468&hit_count=yes&headcode=18&promo=&transfer=&wid=00000000000000000000000000000000

Fax resume to: 970-870-3499 or email resumes to: jobs@ymca.com

Eyecare Specialists is expanding and announces a full-time opening in Craig office for an optician with skills on customer service and retail. You must be detail-oriented, comfortable with technology, enjoy working in fast-paced environment and pass a math aptitude test. Willing to train right person! Benefits, competitive and a fun working atmosphere. Please bring your resume to 1111 W. 9th Way, Suite 110, Craig, CO 81625 or call Julie at 970-824-8132.

Part-time in home Health worker to assist on ventilator. Some medical experience, will train. Fax resume: 879-1616.

▶ Professional

R.H. Robinson & Son, Inc.
Committed to Quality since 1902

We are currently seeking a Supervisor/Foreman/Lead Man. This is potentially a "Long Term, Career Opportunity" and the position is designed for advancement. A certain level of "On the Job" training is expected. Computer skills and Administrative capabilities would be beneficial.

Hayden, CO
295 W. Lincoln Ave., Hayden, CO
970-526-3346

To begin the application process
FwdtoPacket%3E

MANUFACTURER

Credit: TheBITLINK

- What requirement was missing?

Requirements - Importance | TTM

- Time to market must couple with the right product
- Establishing the requirements enables us to agree on and visualize the right product



Credit: TNOK_GARDEN

- What requirement was missing?

IEEE-STD-1220-1998:

- Requirement: a /statement/ that identifies a /product or process operational, functional, or design characteristic or constraint/, which is /unambiguous/, /testable or measurable/, and /necessary for product or process acceptability/ (by consumers or internal quality assurance guidelines).

Definition of Requirements (2)

- **Statement:** a set of traceable, manageable elements identified as requirements.
Requirements can be captured in tabular form, in diagrammatic form in notations such as UML or in domain-specific notations
- **Product or process:** Complete solutions contain varying mixtures of product (things that are built in response to requirements) and process (procedures for using the things that are built).

- **Operational, functional, or design characteristic or constraint:** There are many different kinds of requirement, giving rise to different kinds of language, analysis, modelling, process and solution. Design characteristics cover performance, usability, safety, Maintainability and a host of other qualities.
- **Unambiguous:** A requirement should lend itself to a clear, single understanding, common to all parties involved.
- **Testable or measurable:** Requirements are used to test that the design or solution is acceptable. For this to be possible, the requirement should be quantified, thus providing a means of “measuring” the solution against it.

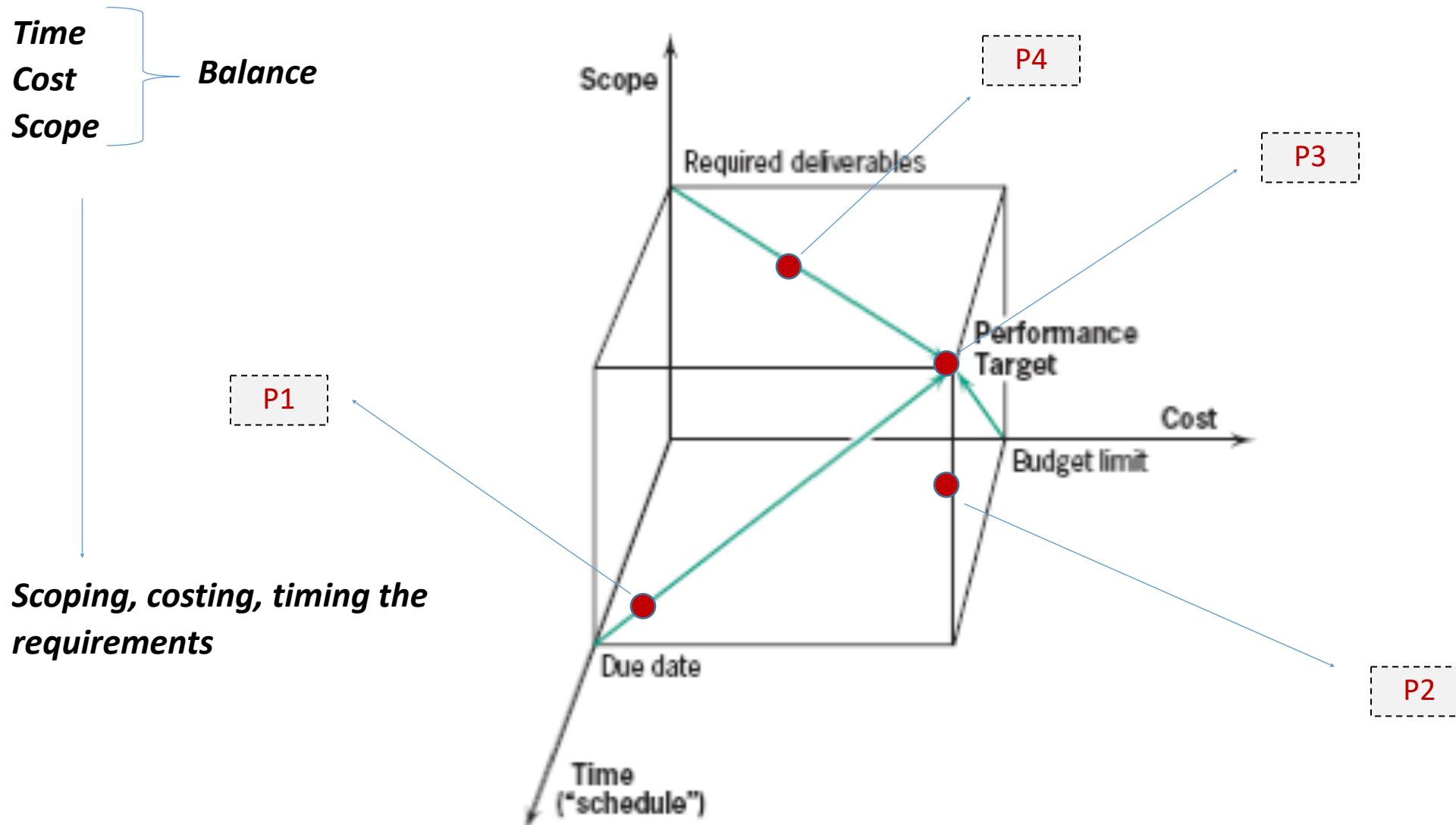
Definition of Requirements (4)

- **Necessary for product or process acceptability:** define how the solution should be tested and accepted. Requirements have an influence in the earliest stages of the development process as well as in the latest stages during acceptance
- **By consumers or internal quality assurance guidelines:** Requirements come from many sources (stakeholders), including but not limited to customers, regulatory bodies, users and internal quality procedures latest stages during acceptance.

- Requirements are essential part of System Engineering Process and Management:
 - They are basis for:
 - Project planning
 - Risk management
 - Acceptance testing
 - Tradeoffs
 - Change control
 - Cost analysis
 - Etc.
- 
- Requirements management

PM vs. Requirements Management

planning, monitoring, analyzing, communicating, and controlling requirements



Is a collection of components which cooperate in an organized way to achieve some desired result → based on REQUIREMENTS

[ISO/IEC 15288]: is a “man-made” or natural combination of elements that are physical and/or abstract and are functioning in harmony. “It may be configured with one or more of the following: hardware, software, data, humans, processes, procedures, facilities, materials and naturally occurring entities”. Systems are classified according to their scope and nature as:

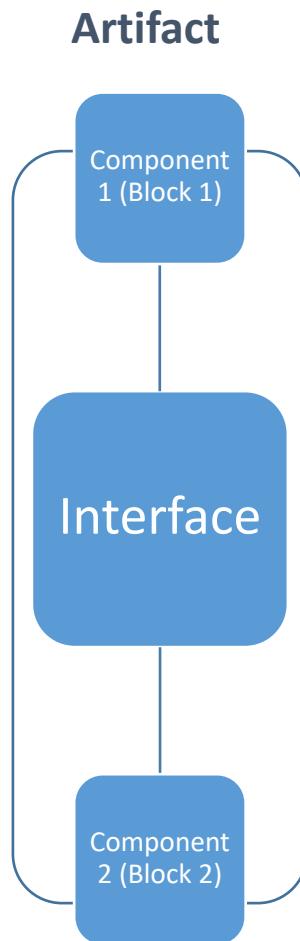
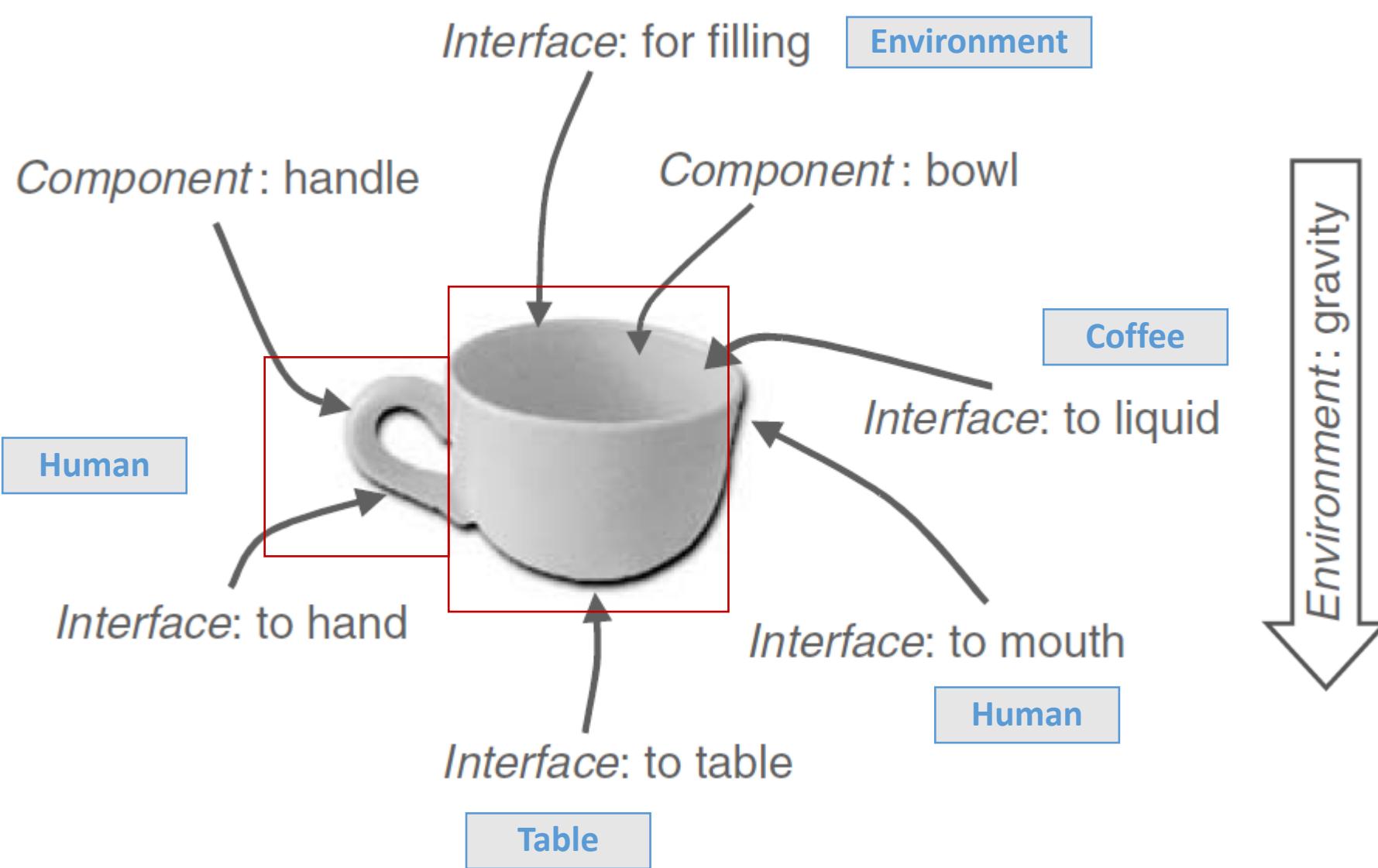
Natural System – “elements, objects or concepts which exist outside of any practical human control. Examples: the real number system, the solar system, planetary atmosphere circulation systems.”

Social System – “elements, either abstract human types or social constructs, or concrete individuals or social groups”.

Technological System – “elements, man-made artifacts or constructs; including physical hardware, software and information”.

SEBOK, INCOSE

Requirements - Example A Cup Viewed as a System



Emergent Properties of a System

- The usefulness of a system does not depend on any particular part of the system, but emerges from the way in which its components interact
- **Emergent properties** may be **desirable**, in that they have been anticipated and designed into the system so as to make the system useful
- **Emergent properties** maybe **undesirable**, unanticipated side effects
- The challenge in systems engineering is to be able to harness desirable emergent properties and avoid the undesirable ones
- Tools to discover emergent properties:
 - Design process
 - VR Simulations
 - HITAL Simulations
 - System use

Cars can be hacked by their tiny, plug-in insurance discount trackers



Karl Koscher and Ian Foster remotely hack a Corvette by tapping into its tracker dongle -- the same used by major insurance companies.

By Jose Pagliery @Jose_Pagliery

Most Popular



Donald Trump calls out Mark Zuckerberg on immigration



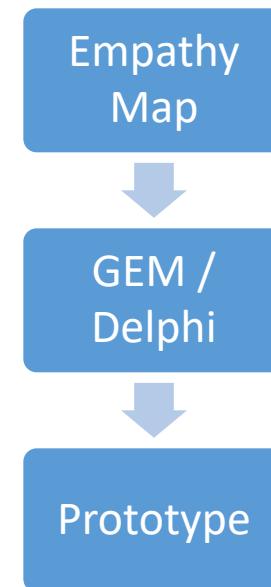
American Apparel warns it may go out of business



UK to test new roads that charge cars as they drive

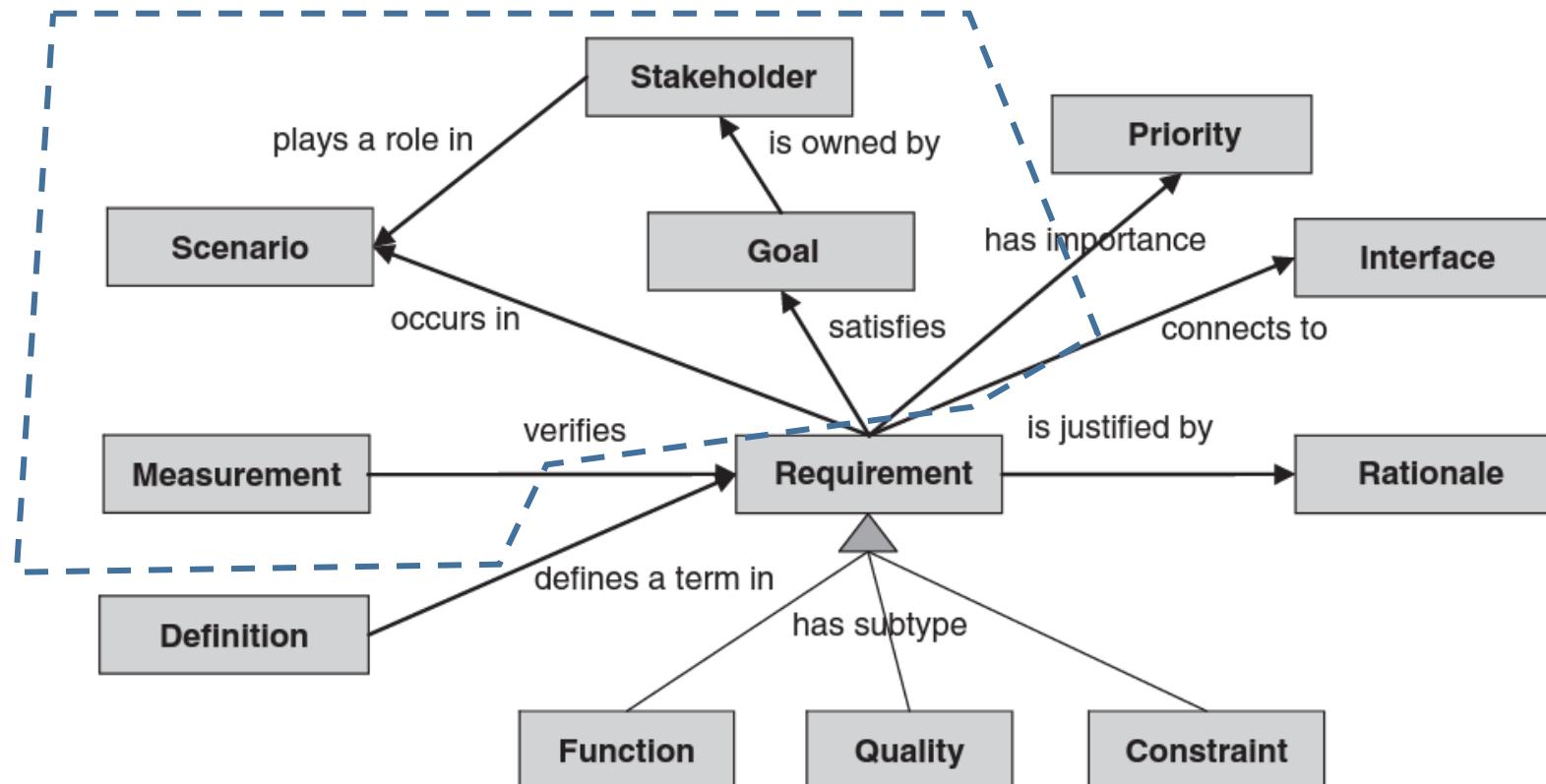
Requirements are created by collaborative work:

- “Discovered, not found”
- “People do not know what they want, it is not enough to ask them”
- Work steadily with the stakeholders towards their goals
- Prototype and test and discover
- Use different techniques



Complementary Views of Requirements

- Requirements cannot be observed or asked for from the users, but have to be created together with all the stakeholders
- People do not know what they want either, so it isn't enough just to ask them
- Use cases do not cover everything



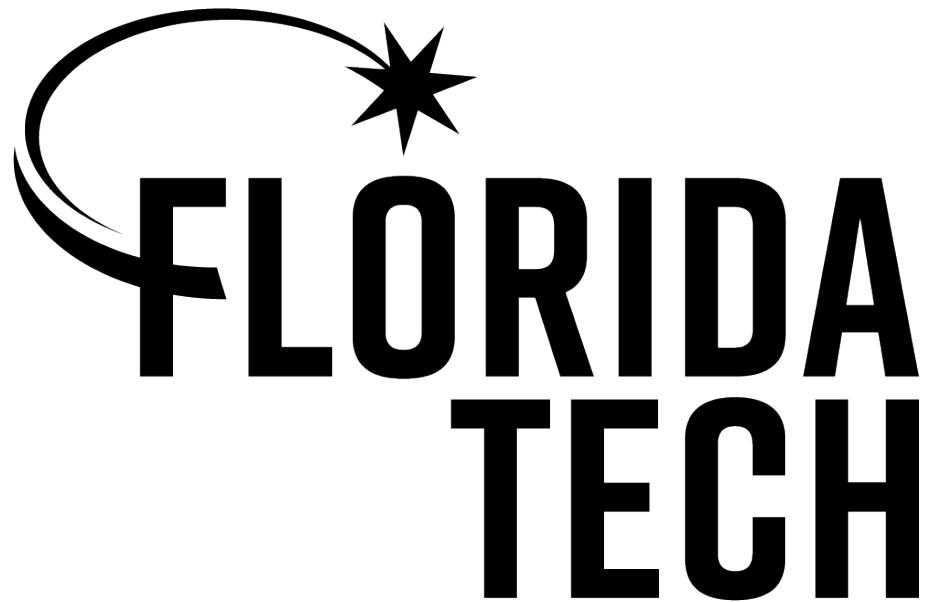
(Alexander,2009)

Questions

Department of Computer & Engineering Sciences

College of Engineering

Florida Institute of Technology

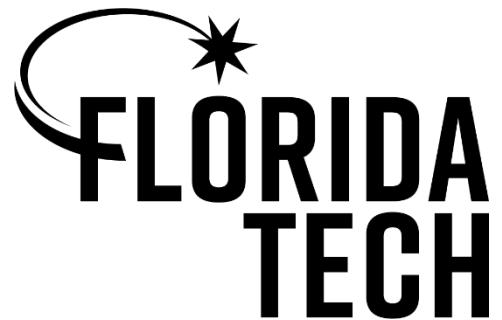


SYS 5460: Stakeholders

Department of Computer & Engineering Sciences

College of Engineering
Florida Institute of Technology

Juan C. Avendano, PhD
Operations Director COES





How the customer explained it



How the project leader understood it



How the analyst designed it



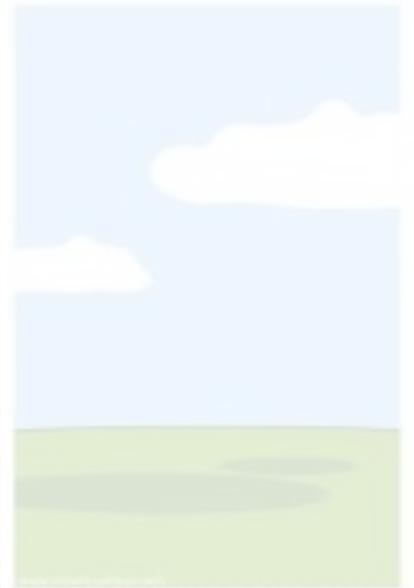
How the programmer wrote it



What the beta testers received



How the business consultant described it



How the project was documented



What operations installed



How the customer was billed



How it was supported



What marketing advertised



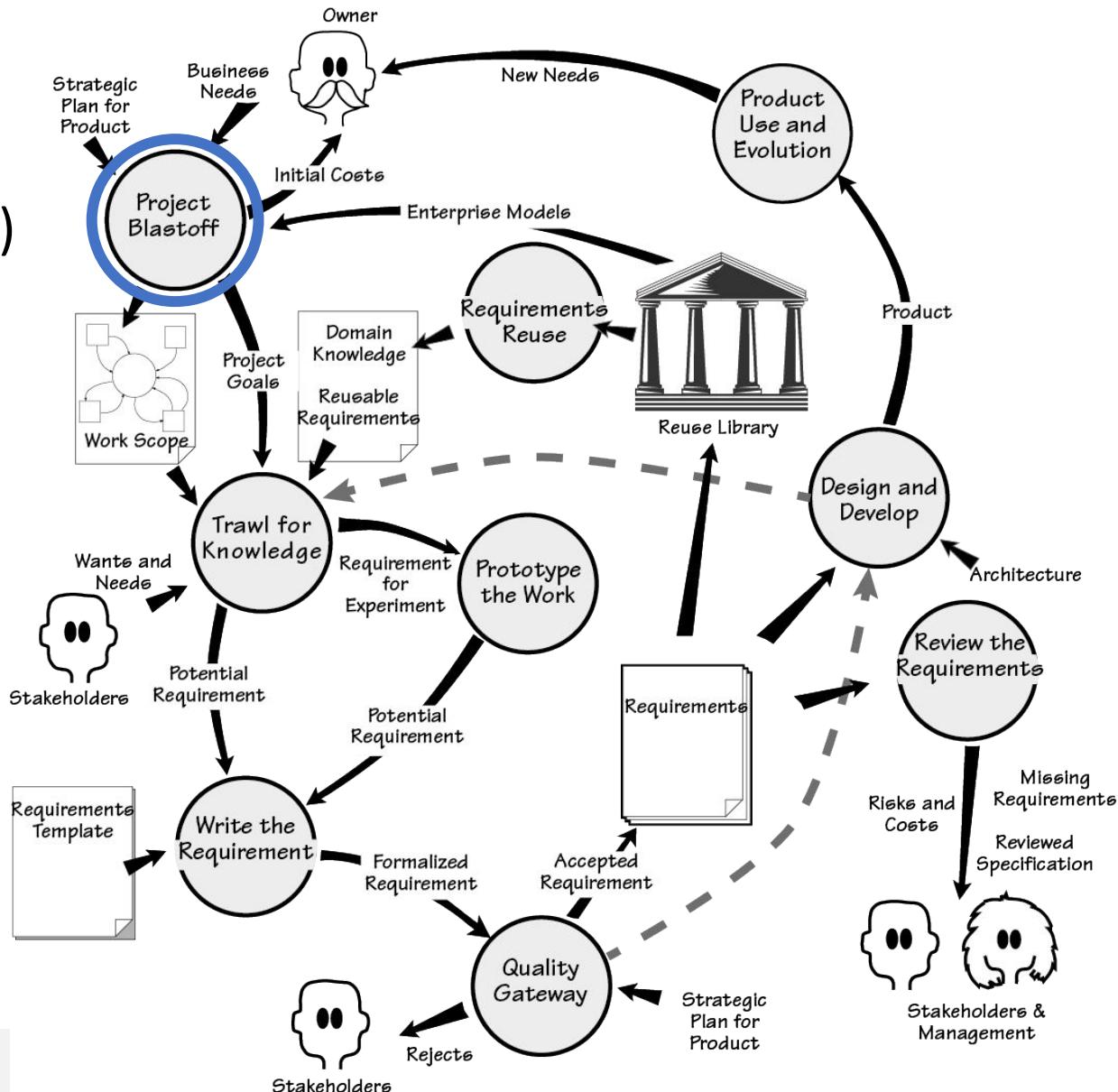
What the customer really needed

Requirements ultimately begin and end with people-Stakeholders (Alexander)

Volere Requirements Process (Robertson)

A requirements elicitation process is recommended

Project Blastoff (Kickoff)



We need to keep in mind the life cycle

Project Blastoff (kickoff)

- The key purpose of project blastoff is to build the foundation for the requirements discovery, and to ensure that all the needed components for a successful project are in place.

- Involvement:

- Principal Stakeholder (sponsor)
 - Key users
 - Lead requirement analyst
 - Technical business experts

- The **sponsor** provides development funding for a project
 - The **champion** provides political support for a project

- Discussions:

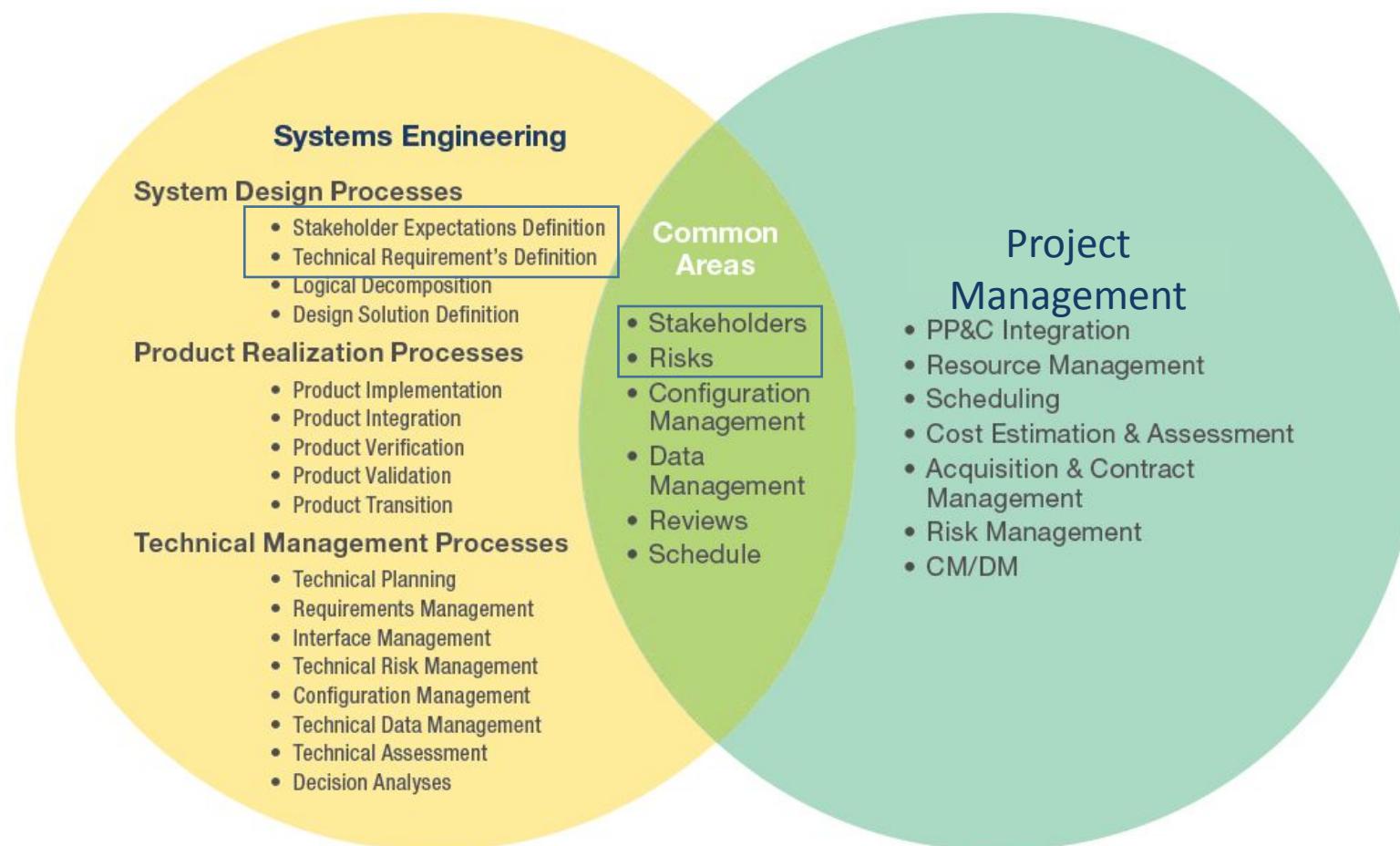
- < Purpose of the project
 - < Scope of the work
 - < Stakeholders
 - < Constraints
 - < Special terminology
 - < Facts & Assumptions
 - < Cost estimates
 - < Risks

- Outcomes:

- < Stakeholders
 - < Project goals
 - < Preliminary assessment of risk and costs
 - < “go/no” go decision

Requirements Process and SE (NASA)

- SE and project management are coupled
- Requirements is part of that intersection



Systems Engineering Processes (NASA)

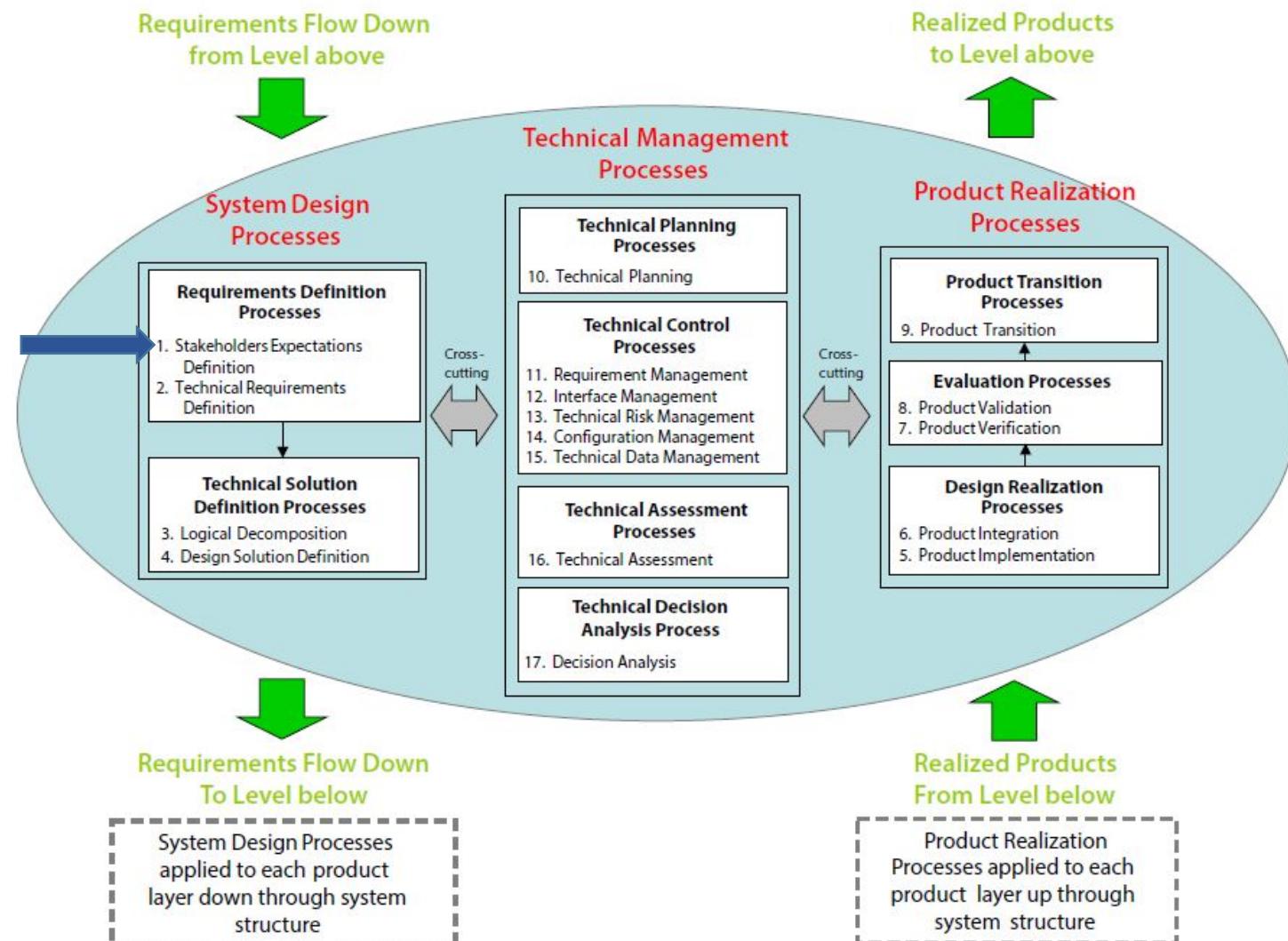


FIGURE 2.1-1 The Systems Engineering Engine (NPR 7123.1)

System Design Process (NASA)

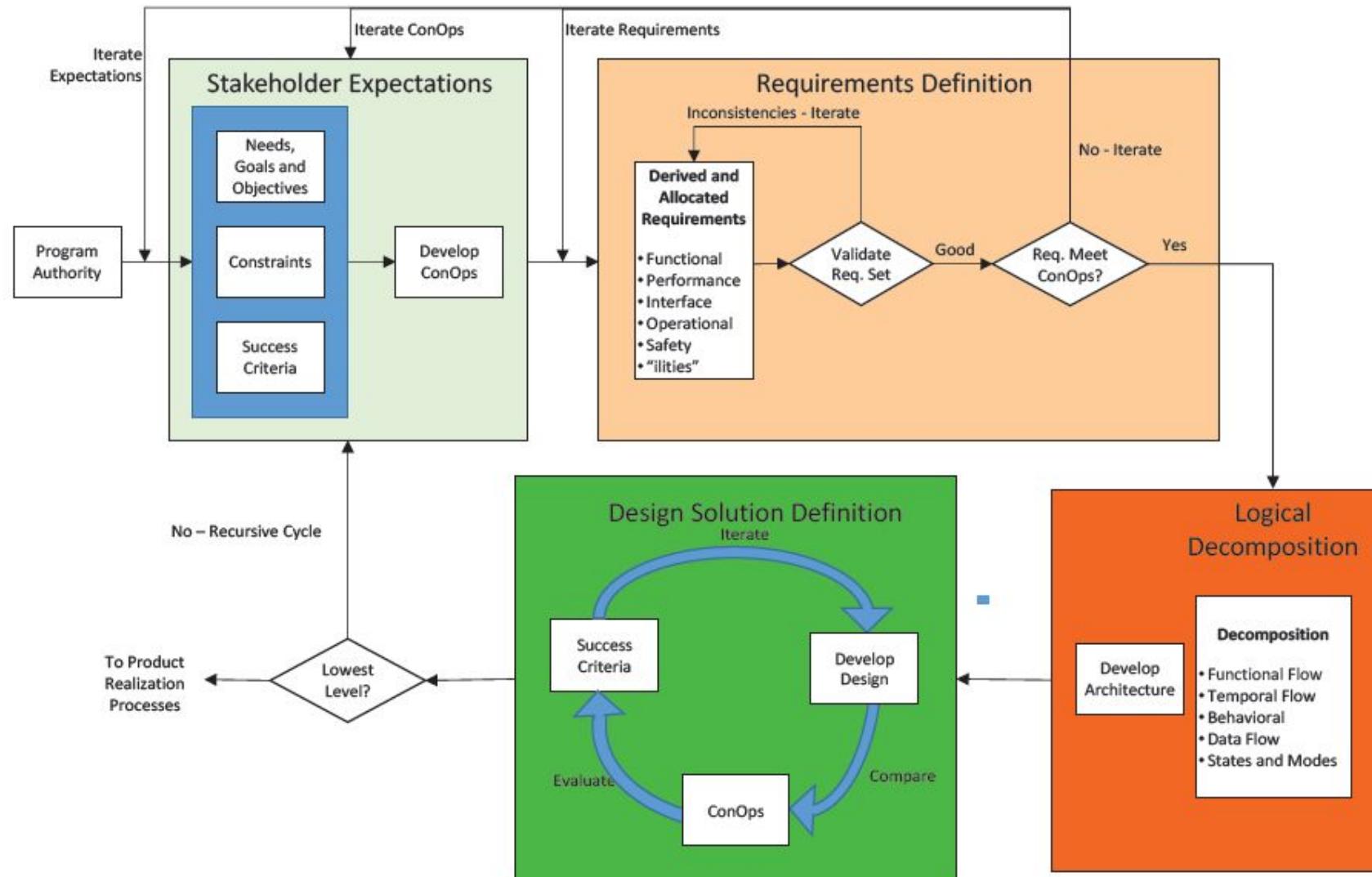


FIGURE 4.0-1 Interrelationships among the System Design Processes

Requirements Engineering and the V-Model (Dick et. al.)

Requirements engineering: the subset of systems engineering concerned with discovering, developing, **tracing**, analyzing, qualifying, communicating and managing requirements that define the system at **successive levels of abstraction** (Dick et. al.)

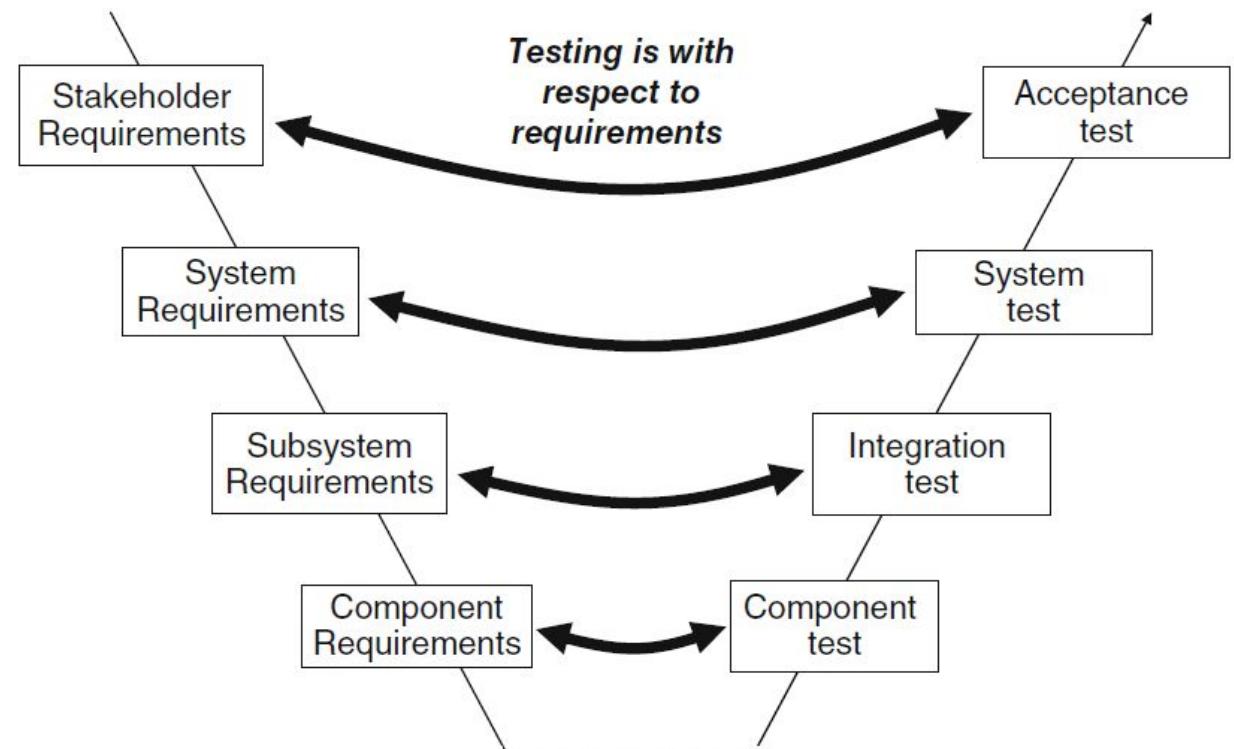


Fig. 1.2 Requirements in the V-Model

Identify: tracing and abstraction

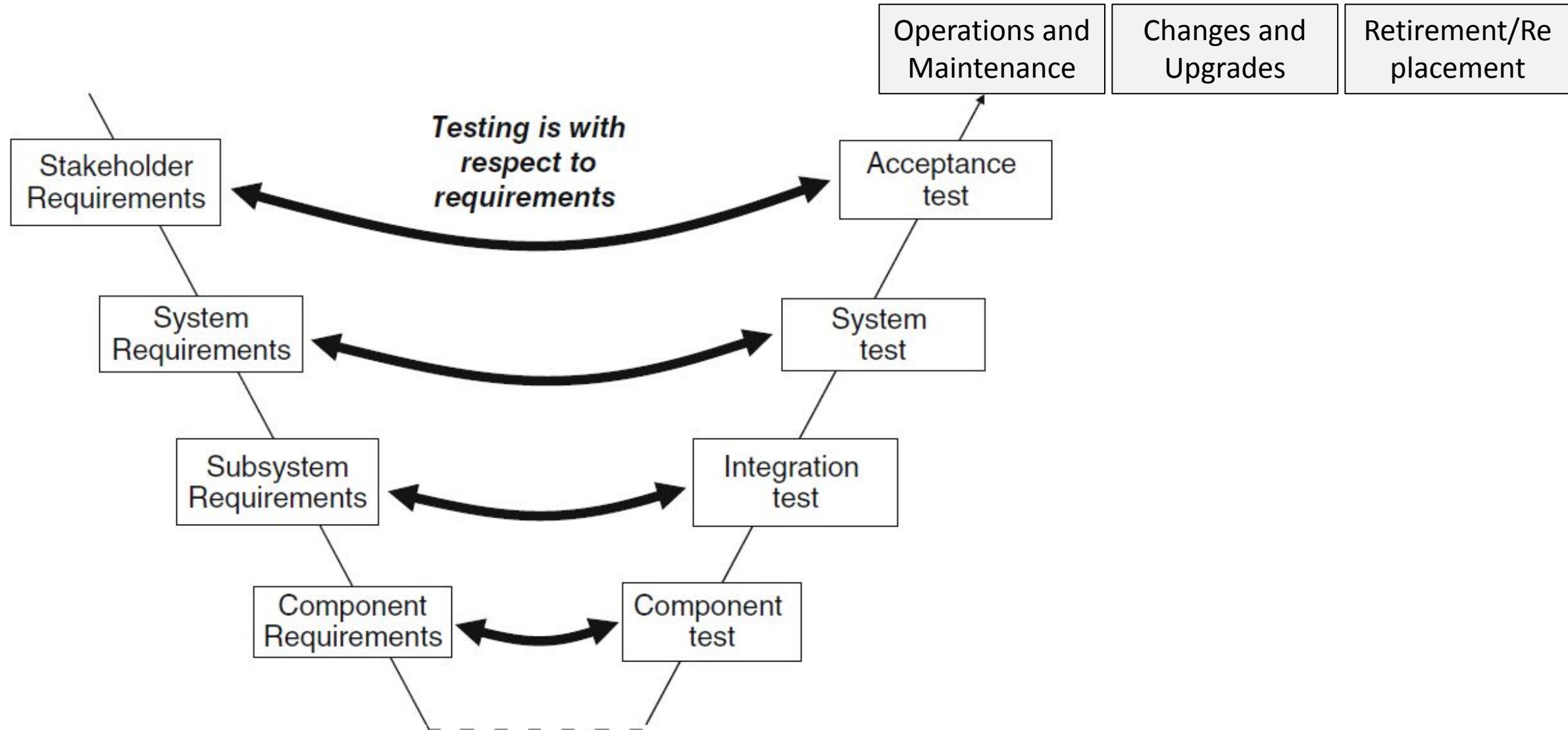
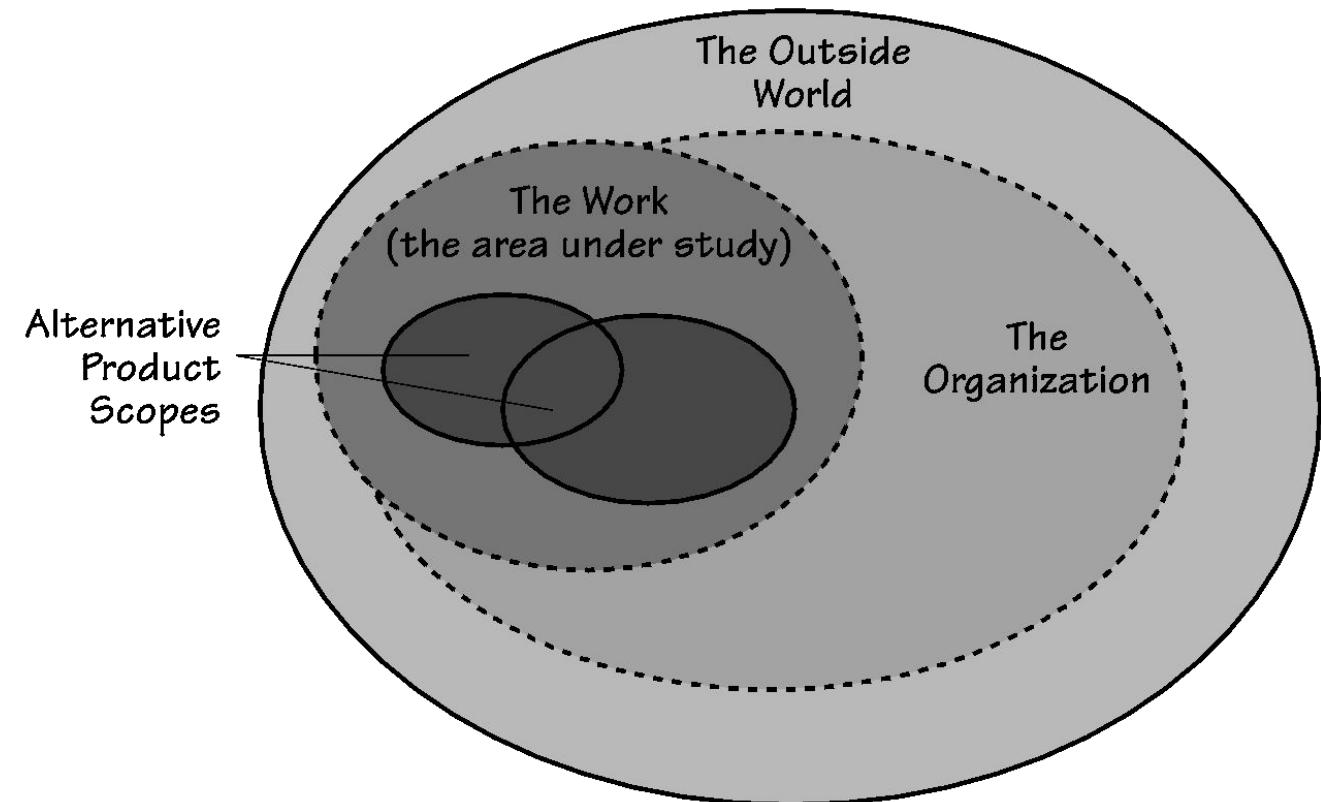


Fig. 1.2 Requirements in the V-Model

System Scope (Robertson)

- The work is the part of the organization that you need to study to discover the requirements.
- The work is usually connected to other parts of the organization and to the outside world.
- You must study the work well enough to understand how it functions.
- This understanding will enable you to come up with alternative scopes for the product and eventually choose the one to build

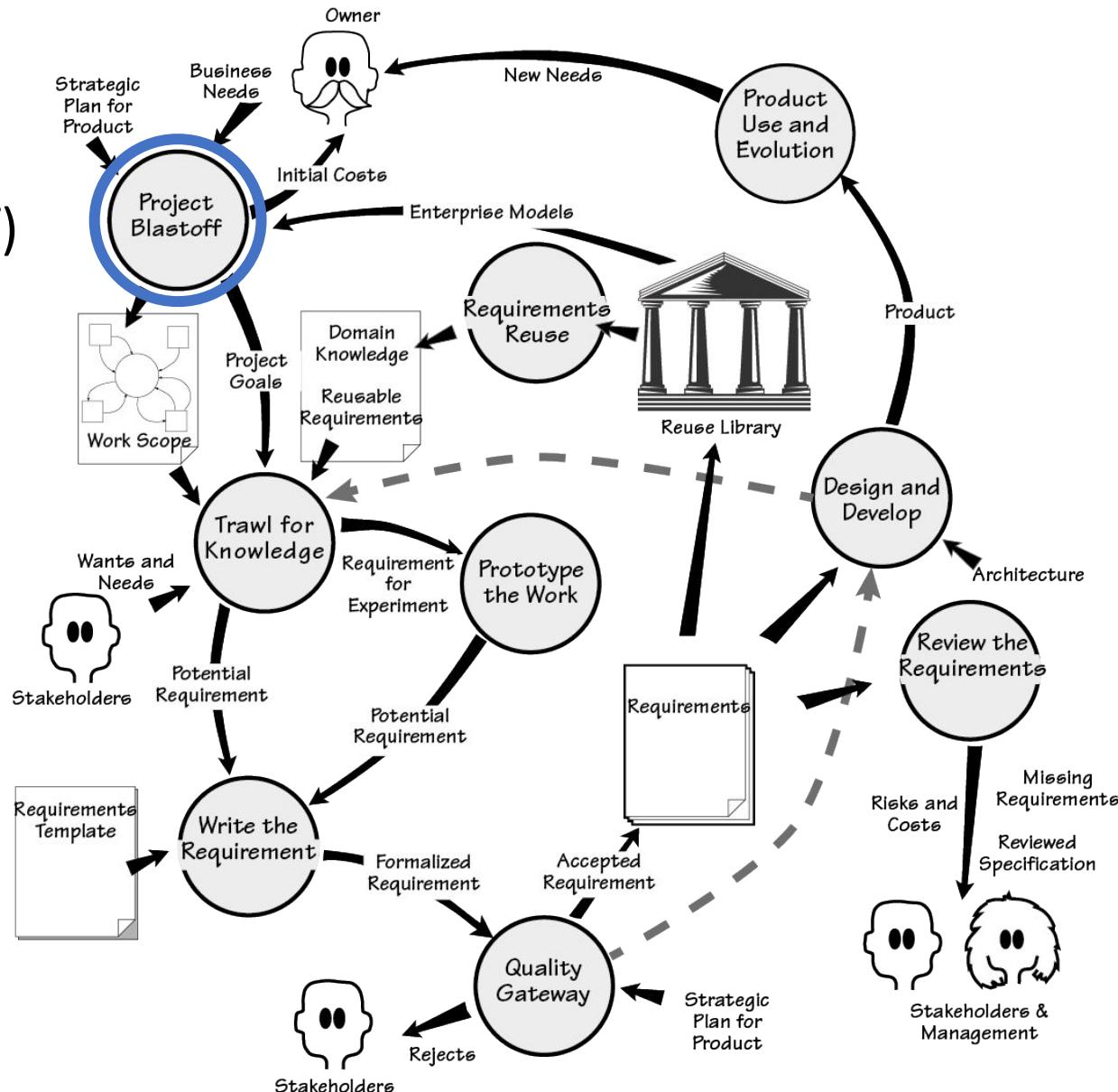


EXERCISE 1

- Think of a product that predicts where ice will form on roads and to schedule trucks to treat the roads with de-icing material
- Be prepared to tell the class what your idea is.

Volere Requirements Process (Robertson)

Project Blastoff (Kickoff)



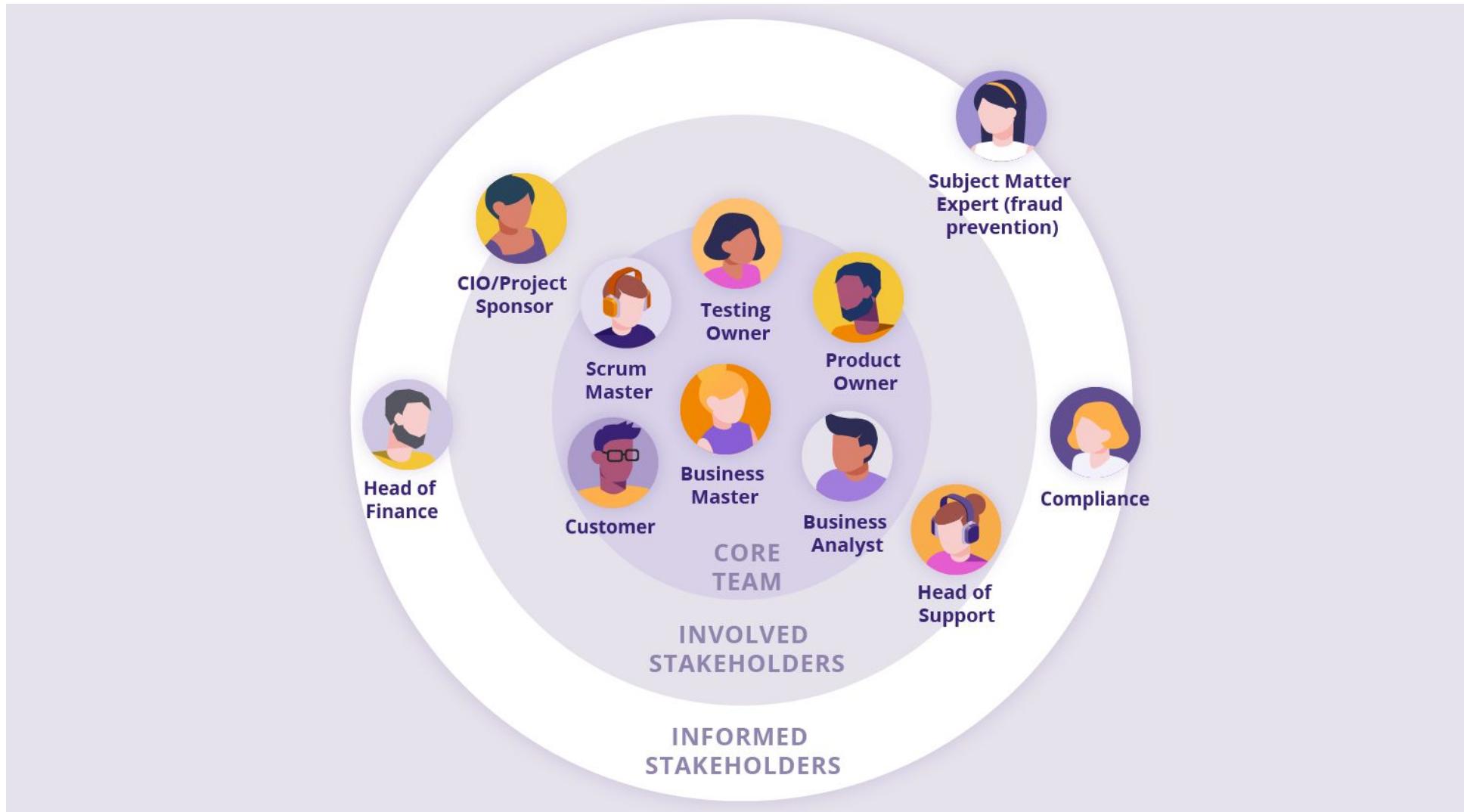
Stakeholders

- **Stakeholder:** An individual, group of people, organization or other entity (system) that has a direct or indirect interest (or stake) in a system
- A stakeholder's interest in a system may arise from (Dick et. al.):
 - **using** the system,
 - **benefiting** from the system (in terms of revenue or other advantage)
 - **being disadvantaged** by the system (in terms, for instance, of cost or potential harm)
 - **being responsible** for the system
- Identify stakeholders by:
 - Asking sponsor or client
 - Examining and organization chart
 - Using a template (onion model)
 - Comparison with similar projects
 - Analyzing the context of the project

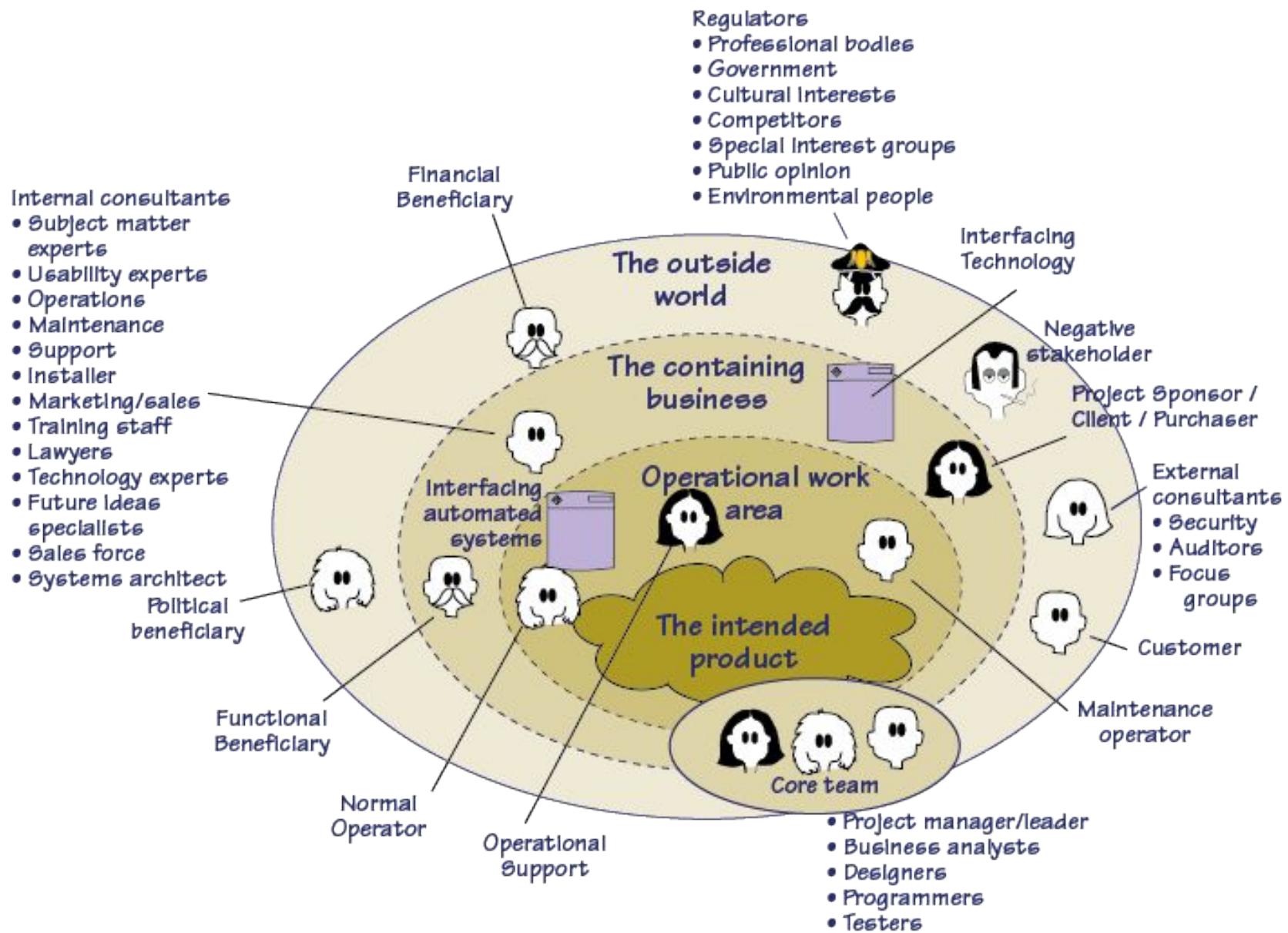
Stakeholder (Cont)

- Identify Stakeholders. ...
- Analyze Stakeholders. ...
- Map Stakeholders. ...
- Prioritize Stakeholders. ...
- Be Inclusive. ...
- Communicate Clearly. ...
- Be Open and Honest. ...
- Remain Available.

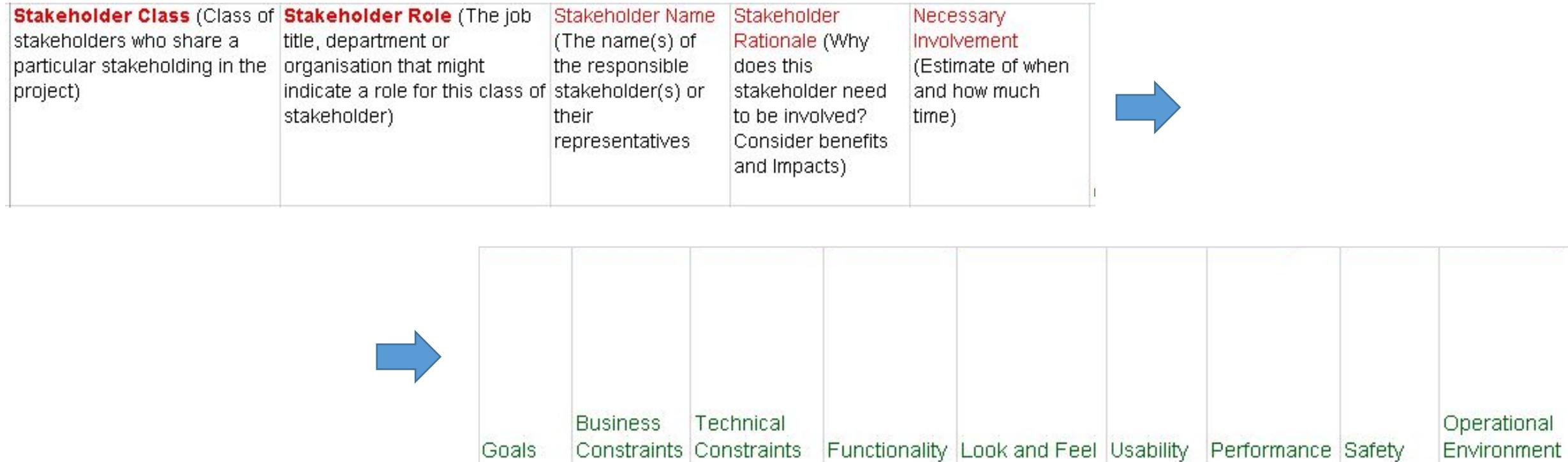
Stakeholder Map



Example: IceBreaker Stakeholders

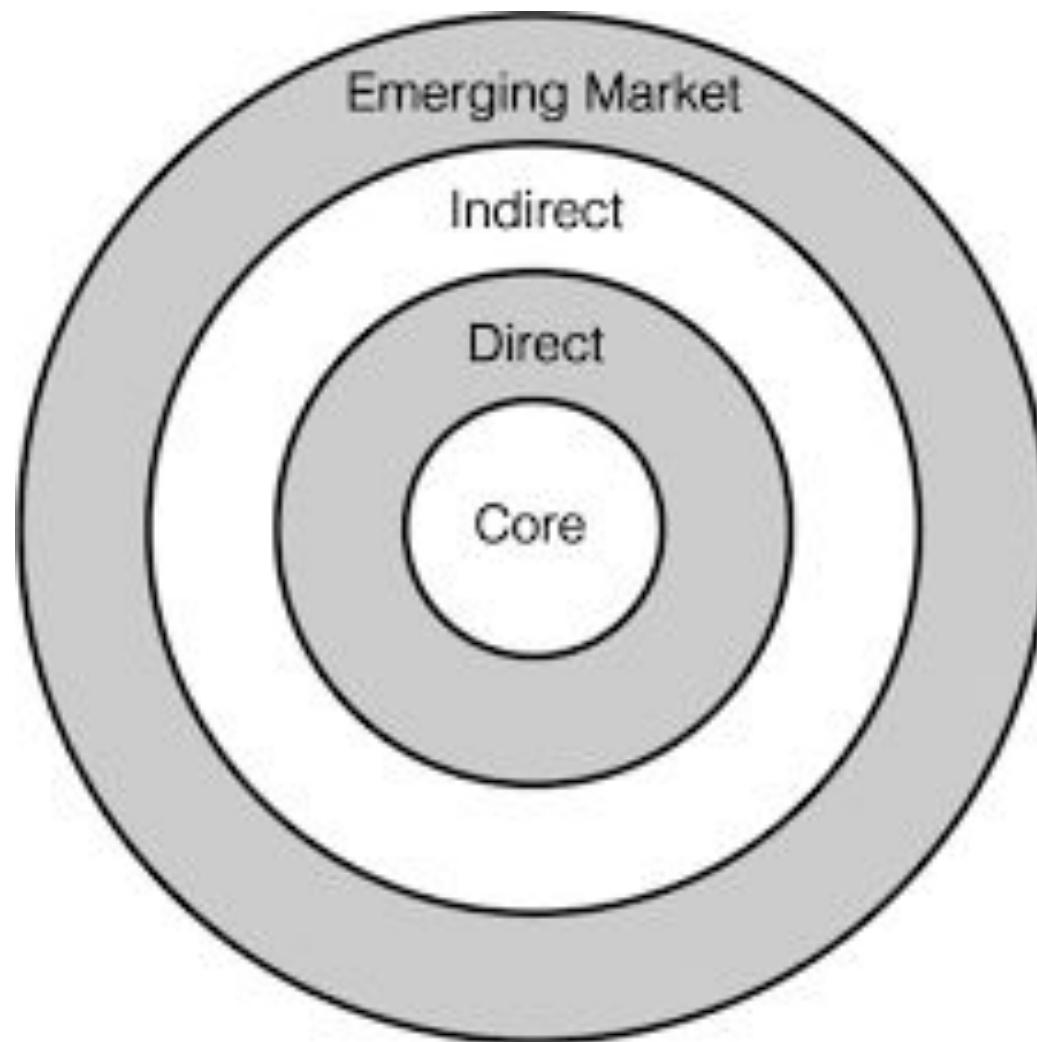


Excel Template (Robertson)



EXERCISE 2

- In your groups, Identify as many stake holders as possible for the proposed system
- Identify as many interactions as possible
- Prepare to share with the class



Surrogate Roles

- Many roles including paid work are surrogates (on behalf of someone else):
 - Company director represents shareholders
 - Lawyer represents a plaintiff
 - User interface designer represents human operator
- In some cases the requirements engineer may not be able to talk to the actual users (large multi-year projects)
- Multiple groups of user for the product
- **Surrogacy can be dangerous** if people are wrong about the needs and whishes of the people they claim to represent



CBS NEWS / March 19, 2019, 7:11 AM

FAA approval process relies
on Boeing, other
manufacturers to self-police

Operational Roles in the System

| Stakeholder role | Type of requirement | Discovery technique |
|-----------------------------------------------------------|------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Normal operators (possibly in many different roles) | Scenarios (Chapter 5) Usability (Chapter 6) | |
| Interfacing | Interface definitions (Chapter 4) | Interview (Chapter 10) Apprenticing (Chapter 10) Observation (Chapter 10) Workshops (Chapter 11) Data modelling (Chapter 8) Prototyping (Chapter 12) Archaeology (Chapter 12) |
| Maintenance | Maintenance functions/scenarios (Chapter 5) Diagnostics, built-in test | |
| Support | Support functions | |
| Functional beneficiary | Product functions/scenarios (Chapter 5) Performance targets (Chapter 9) | |
| Financial beneficiary | Mission, objective (Chapter 3) | Interview (Chapter 10) Read policy documents |
| Regulator | Regulations, laws, standards, guidance Responses to safety case, compliance statements, etc | Legal advice on regulations, etc Negotiate compliance |
| Experts, specialists in disciplines | Safety, security, reliability, usability, etc (Chapter 6) Constraints (Chapter 6) | Analysis, simulation, modelling, standards |
| Manufacturer | Producibility | Interview (Chapter 10) Workshop (Chapter 11) Prototyping (Chapter 12) |
| Marketing (surrogate, on behalf of mass-market customers) | Mass market (consumer) Preferences by group (age, income, etc) | Market survey, Field trials Observation (Chapter 10) Prototyping (Chapter 12) Analogous products (Chapter 12) Competitor analysis |
| Product manager, purchaser | Priorities Programme, schedule Budget (cost) | Prioritisation (Chapter 13) Trade-offs (Chapter 14) |
| The public | (Lack of negative impact) | Public meetings, Focus groups, Consultation, Roadshows |

- Requirements can be discovered by the analyst based on existing systems or mathematical modeling
- Other requirements are coming from stakeholders
- The most likely roles that may contribute with the majority of the requirement is listed in the table

Deriving Requirements from Stakeholders

- Goal and scenario modeling are useful tools to obtain requirements from stakeholders
- Surveys, public meetings are the recommended tools for non-operational stakeholders

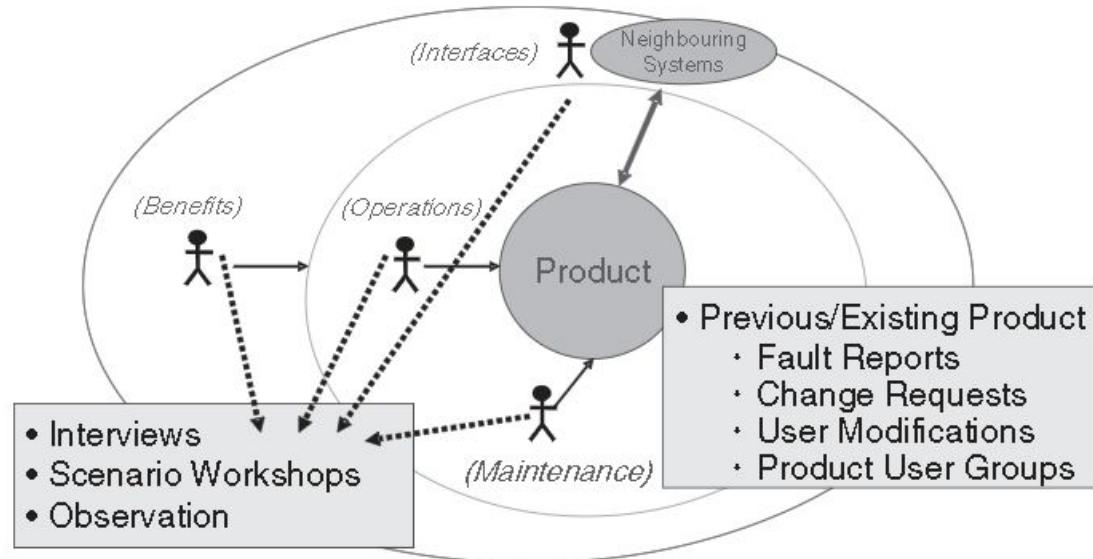


Figure 2.6: Requirements from operational stakeholders.

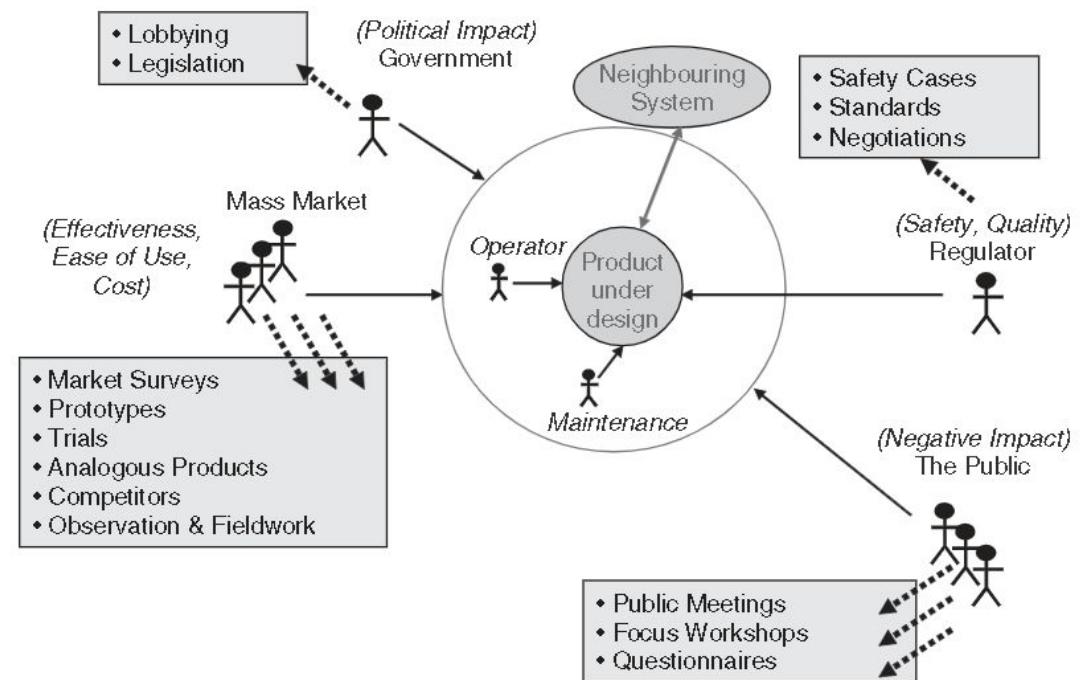


Figure 2.7: Requirements from non-operational stakeholders.

Goals

- A goal is something that the stakeholder wants to achieve
- Goals are permitted to be neither fully achievable nor measurable (at the beginning)
- Requirements are achievable and measurable
- Goals!=requirements
- Requirements analysis function refines the goals into realistic and measurable target
- Goals may be in conflict (examples)
- Project without goals are vulnerable to pressure to add requirements
- Unstated conflicts lead to tension and confusion
- Stated conflicts can be used in tradeoff analyses

| (Stakeholder) goals | (Product) requirements |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Belong to different stakeholders | Agreed by all |
| May conflict, indicating design trade-offs; these often drive project design activity and the choice of life cycle (e.g. iterations with analysis or prototypes of competing options, to reduce risk) | Must not conflict in the chosen technology; therefore, design envelope must be known to a sufficient degree (leaving as much freedom inside that envelope as possible) |
| May be an ideal, unattainable, indicating what is hoped for | Must be realisable within limits of budget, timescale, technology, and skill available |

Goals

- Definition
- Goal diagrams
- Goal conflicts
- Examples

- Highest level ,large scale goals are mission statements
- Long lived goals are usually policies
- Low-level goals are functions

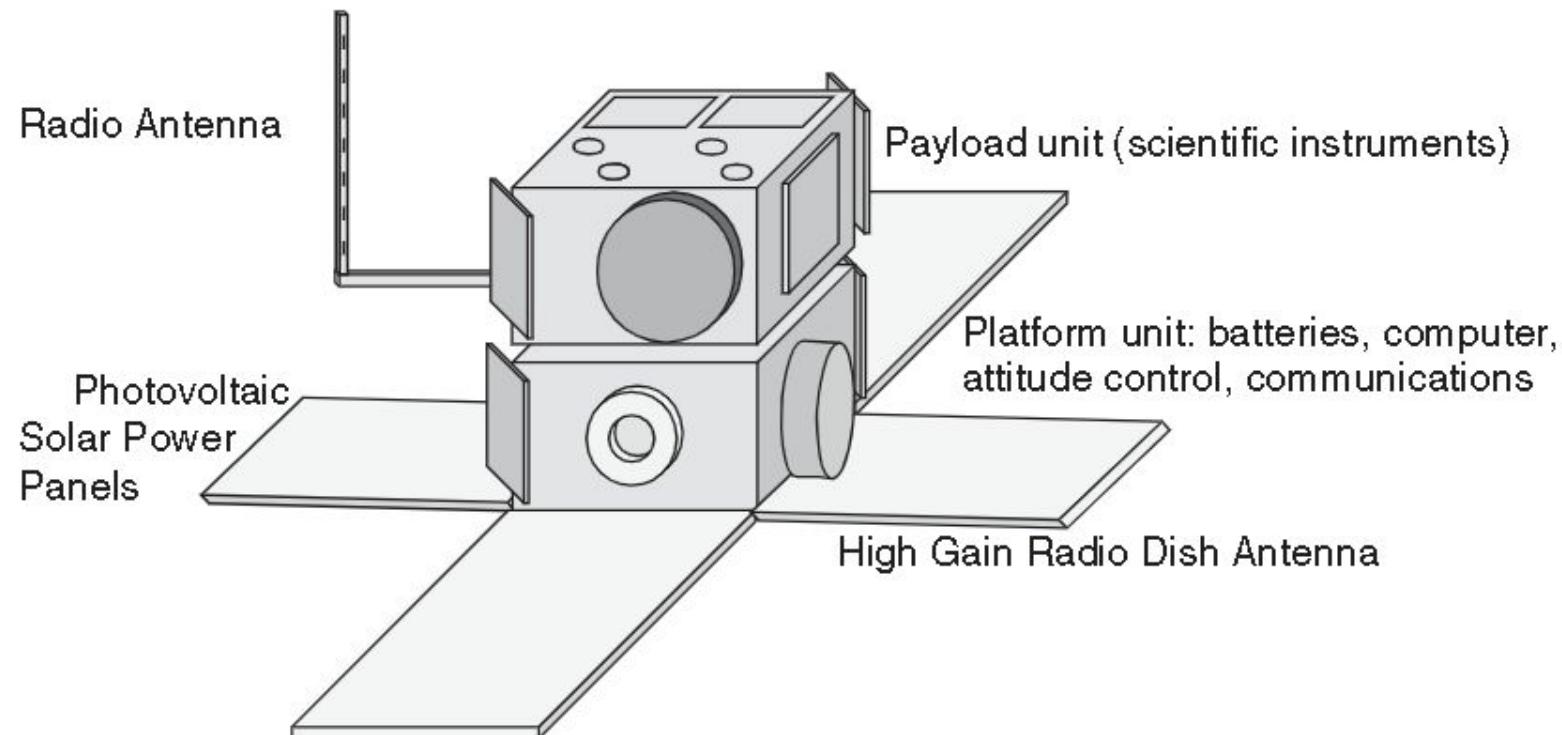
- Goals turn into requirements when:
 - Fully verifiable
 - Prioritized

Types of Goals

- Goals can be functional or quality
- Functional is to do something (carry passengers)
- Quality goal refers to the way something is done (e.g. comfort)

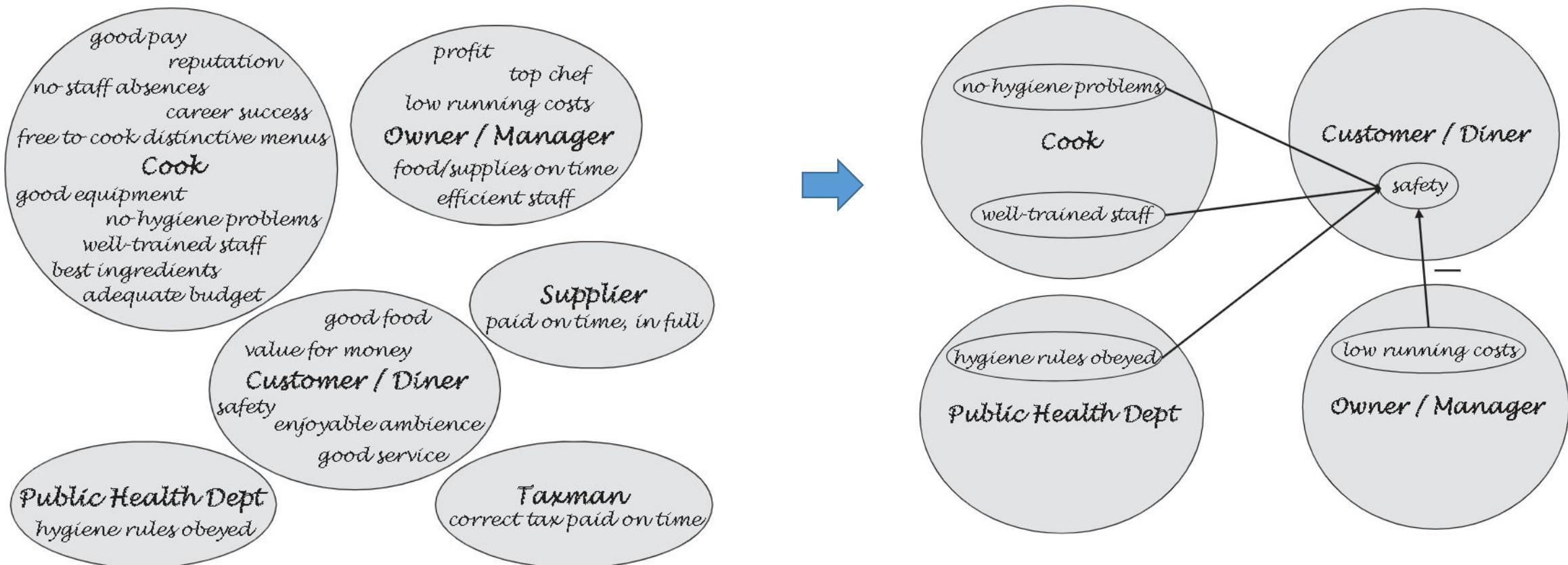
Example: Spacecraft

- To fly in space
- To do worthwhile science
- To send the results back to earth



Example: Restaurant

| Role | Cook | Owner/ manager | Customer/ diner | Supplier | Public health department | Taxman |
|-------|----------------------------------------------------|------------------------------------------------|-------------------------------------|-----------------------|--------------------------|--------------------------|
| Goals | Good pay Reputation No staff absences ... | Profit Top chef Low running costs ... | Good food Value for money ... | Paid on time, in full | Hygiene rules obeyed | Correct tax paid on time |
| | | | | | | |



Tips for Discovering Goals

- Start with the people you know
- Identify stakeholders
- Use interviews and workshops to find out what their goals are
- Listen carefully for signs that goals may in conflict
- For specific feature request , ask why?
- Play back what you have heard to confirm that your interpretation is accurate

Stakeholders in AGILE

Stakeholders in Agile Development: Personas

- Story about an invented person that includes:
 - Name
 - Age
 - Job
 - Family
 - Hobbies
 - Residence
 - Favorite food
 - Attitude towards technology, money
 - Include a graphical representation
- Summarize persona in a profile



Who: Make a Persona

“Mary”



Demographics

- Working mom
- 34 years old
- Lives in Reading, works in London
- Married, 2 kids
- Household 125k/yr

Behaviors

- Has a housecleaner
- Buys take-away 3 nights/wk
- Frequently feels overwhelmed when she “forgets” something

Needs & Goals

- Help! Running errands, managing kids, keeping things running
- Time for her girlfriends
- To feel like she “has it sorted”
- “To clone herself”

Excerpt from Luxr.co: <http://www.slideshare.net/clevergirl/luxr-oneday-workshop>

Product Vision

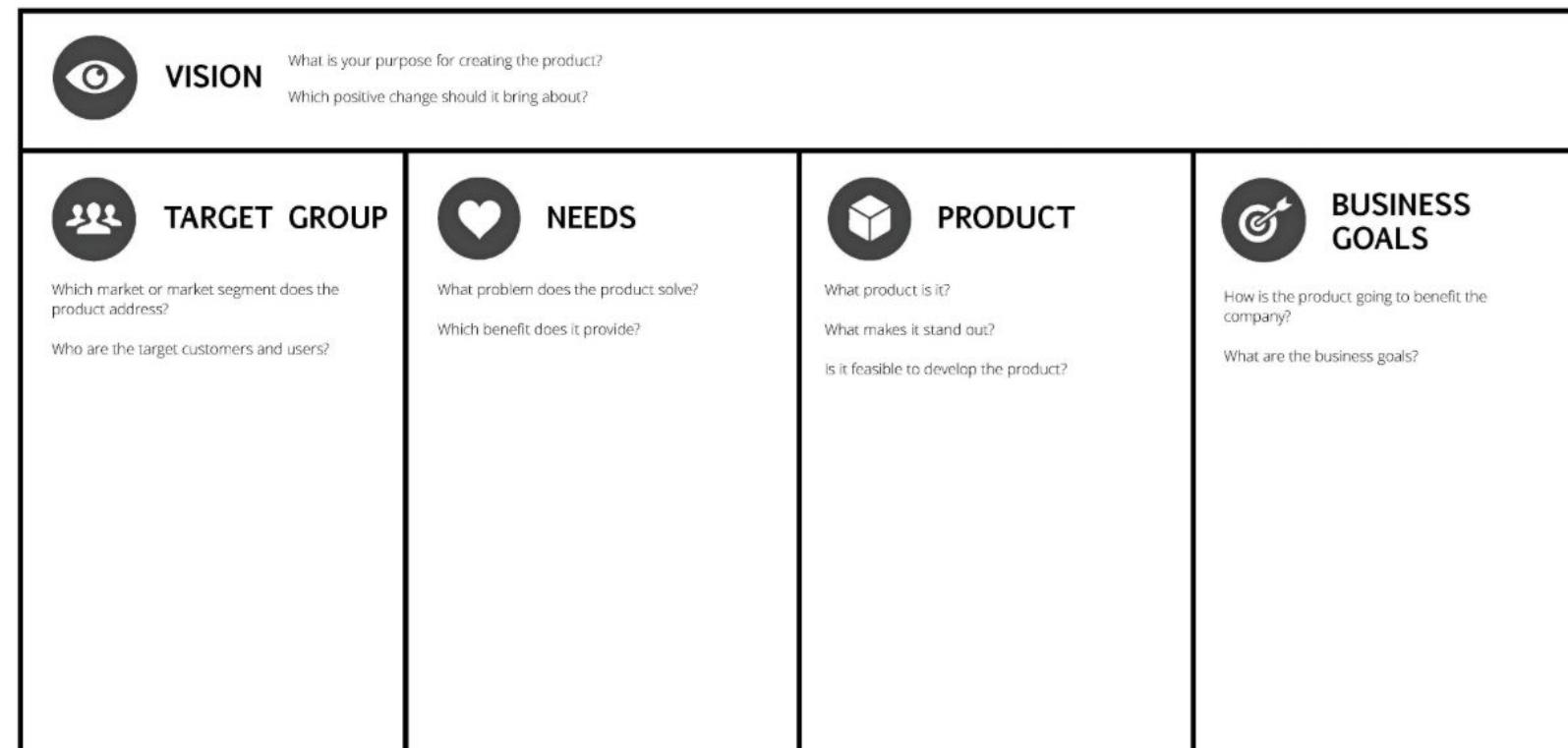
- Create a product vision
 - Use a product vision board template

Vision Tips:

- **For:** Stakeholders
- **That:** Needs , goals
- **The:** Product
- **Is A:** Description
- **Which:** Function main benefit
- **Differently:** Than current methods/products
- **Our Product:** Additional Benefits

THE PRODUCT VISION BOARD

 romanpichler



EXERCISE 3

- In your groups, identify a product that can be completed within the scope of this class
- Product may be software, or hardware (3D printed Gadgets, small electronic assembly, etc.)
- Create a product vision board
- Prepare a 3 min presentation to share with the group
- You have 15 min

References Used in this Lecture

- [1] S. Robertson and J. Robertson, *Mastering the requirements process: getting requirements right*, 3rd ed. Upper Saddle River, NJ: Addison-Wesley, 2013.
- [2] Alexander, *Discovering requirements*. 2009.
- [3] J. Dick, E. Hull, K. Jackson, and SpringerLink (Online service), *Requirements engineering*, 4th;Fourth; Cham: Springer, 2017.
- [4] “ISO/IEC/IEEE International Standard - Systems and software engineering -- Software life cycle processes,” *IEEE STD 12207-2008*, no. Generic, pp. 1–138, 2008, doi: 10.1109/IEEESTD.2008.4475826.
- [6] S. NASA, “NASA systems engineering handbook,” *Natl. Aeronaut. Space Adm. NASASP-2007-6105 Rev2*, 2016.

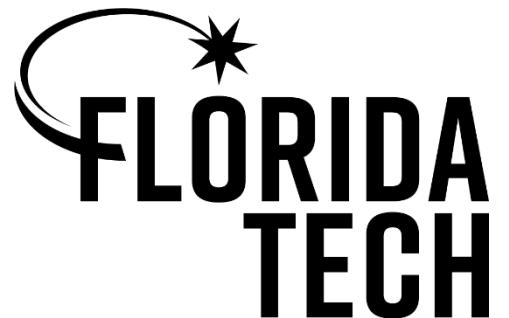
SYS 5460: Context , Interfaces, Scope

Contents

- Concept of Operations
- System Context
- Interfaces
- Examples

**Department of Computer &
Engineering Sciences**

College of Engineering
Florida Institute of Technology





How the customer
explained it



How the project leader
understood it



How the analyst
designed it



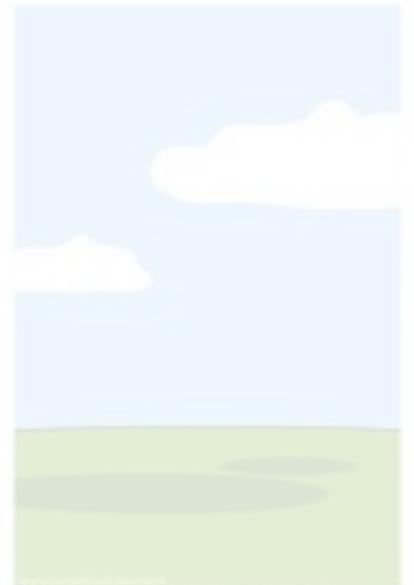
How the programmer
wrote it



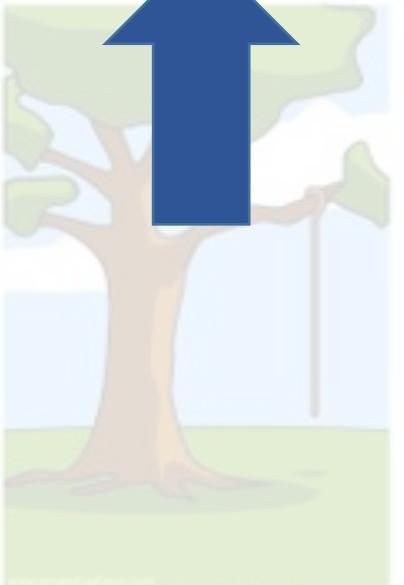
What the beta testers
received



How the business
consultant described it



How the project was
documented



What operations
installed



How the customer was
billed



How it was supported



iSwing

What marketing
advertised

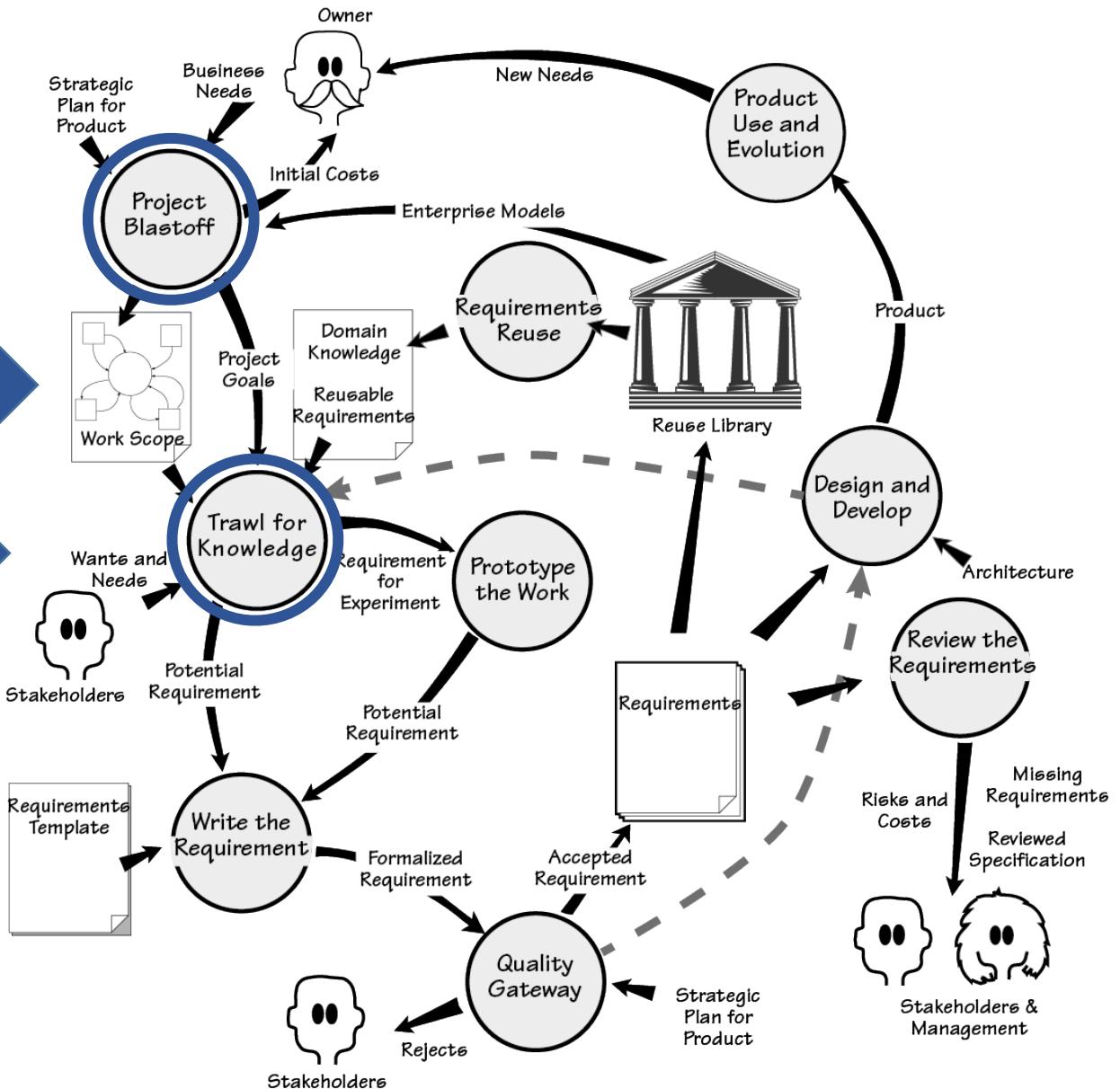


What the customer
really needed

Volere Requirements Process (Robertson)

Model system functionality

Concept of Operations



Standards For Concept of Operations

Previous standards

ANSI/AIAA-G-043 Outline

1. Scope
2. Referenced Documents
3. User-Oriented Operational Description
4. Operational Needs
5. System Overview
6. Operational Environment
7. Support Environment
8. Operational Scenarios

IEEE 1362 Outline

1. Scope
2. Referenced Documents
3. The Current System or Situation
4. Justification for and Nature of Changes
5. Concepts for the Proposed System
6. Operational Scenarios
7. Summary of Impacts
8. Analysis of the Proposed System



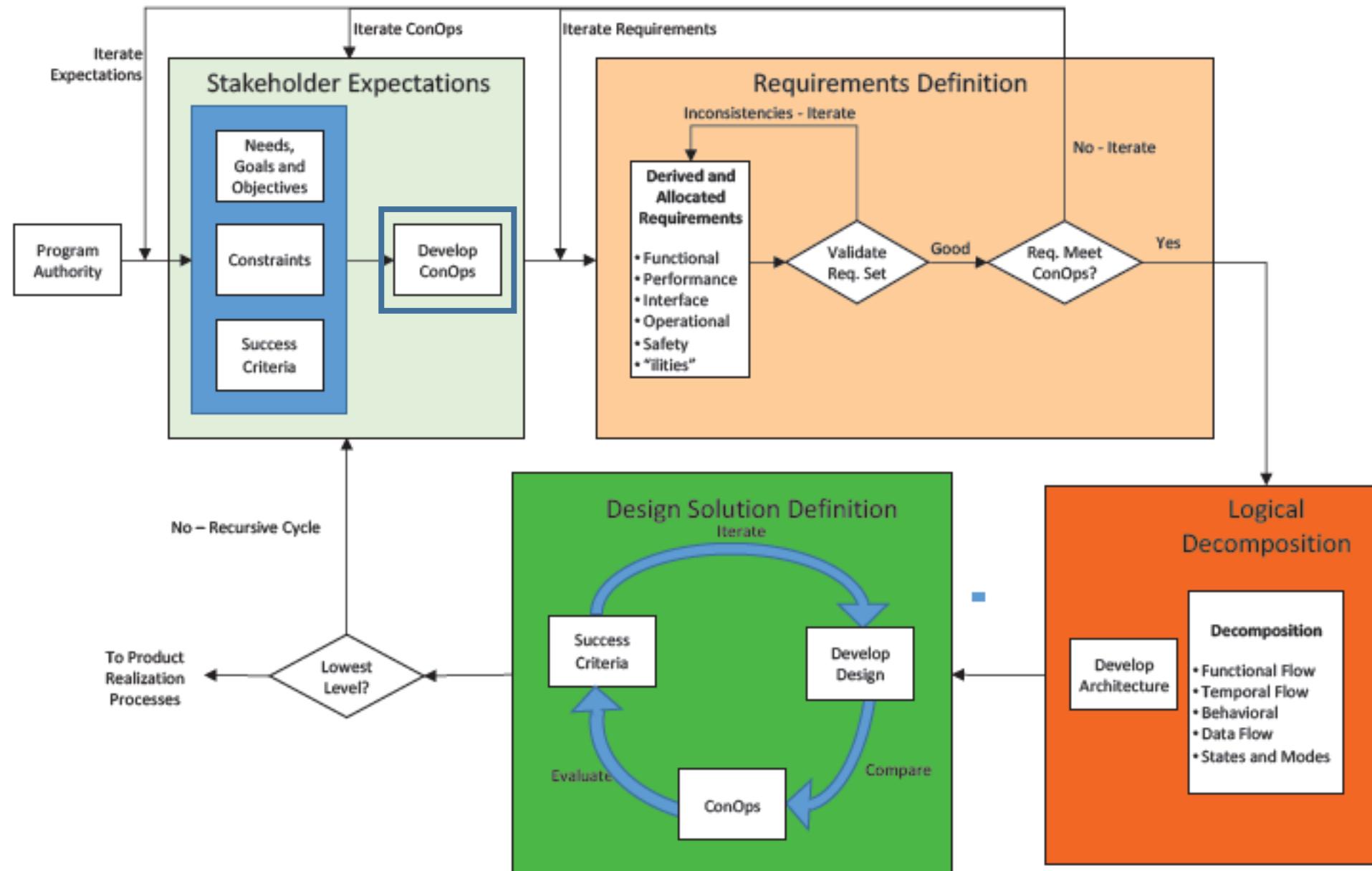
Current standards ISO/IEC 12207 Software lifecycle process or ISO/IEC 15288 Systems and Software lifecycle processes

concept of operations

verbal and/or graphic statement, in broad outline, of an organization's assumptions or intent in regard to an operation or series of operations

Note 1 to entry: The concept of operations frequently is embodied in long-range strategic plans and annual operational plans. In the latter case, the concept of operations in the plan covers a series of connected operations to be carried out simultaneously or in succession. The concept is designed to give an overall picture of the organization operations. See also operational concept.

Note 2 to entry: It provides the basis for bounding the operating space, system capabilities, interfaces and operating environment.



DoDAF Viewpoints and Models

OV-1: High-Level Operational Concept Graphic

OV-2: Operational Resource Flow Description

OV-3: Operational Resource Flow Matrix

OV-4: Organizational Relationships Chart

OV-5a: Operational Activity Decomposition Tree

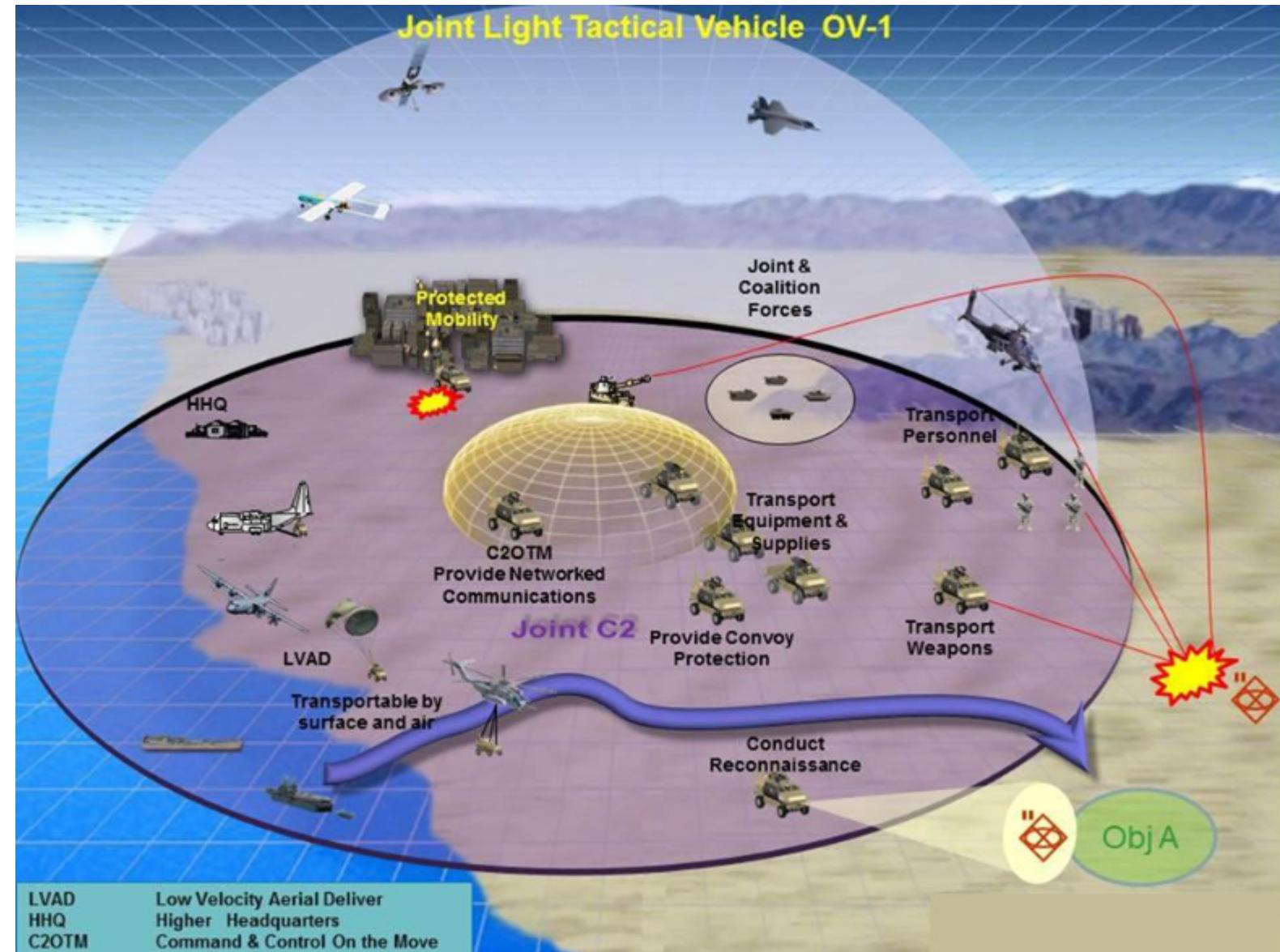
OV-5b: Operational Activity Model

OV-6a, 6b, 6c: Introduction

OV-6a: Operational Rules Model

OV-6b: State Transition Description

OV-6c: Event-Trace Description



Concept of Operations Document (IEEE 1362-1998)

Title page

Revision chart

Preface

Table of contents

List of figures

List of tables

1. Scope

 1.1 Identification

 1.2 Document overview

 1.3 System overview

2. Referenced documents

3. Current system or situation

 3.1 Background, objectives, and scope

 3.2 Operational policies and constraints

 3.3 Description of the current system or situation

 3.4 Modes of operation for the current system or situation

 3.5 User classes and other involved personnel

 3.6 Support environment

4. Justification for and nature of changes

 4.1 Justification of changes

 4.2 Description of desired changes

 4.3 Priorities among changes

 4.4 Changes considered but not included

5. Concepts for the proposed system

 5.1 Background, objectives, and scope

 5.2 Operational policies and constraints

 5.3 Description of the proposed system

 5.4 Modes of operation

 5.5 User classes and other involved personnel

 5.6 Support environment

6. Operational scenarios

7. Summary of impacts

 7.1 Operational impacts

 7.2 Organizational impacts

 7.3 Impacts during development

8. Analysis of the proposed system

 8.1 Summary of improvements

 8.2 Disadvantages and limitations

 8.3 Alternatives and trade-offs considered

9. Notes

Appendices

Glossary

ConOps

Project Drivers

1. The Purpose of the Project
2. The Stakeholders

Project Constraints

3. Mandated Constraints
4. Naming Conventions and Terminology
5. Relevant Facts and Assumptions

Functional Requirements

6. The Scope of the Work
7. The Business Data Model and Data Dictionary
8. The Scope of the Product
9. Functional Requirements

Non-functional Requirements

10. Look and Feel Requirements
11. Usability and Humanity Requirements
12. Performance Requirements
13. Operational and Environmental Requirements
14. Maintainability and Support Requirements
15. Security Requirements
16. Cultural Requirements
17. Legal Requirements

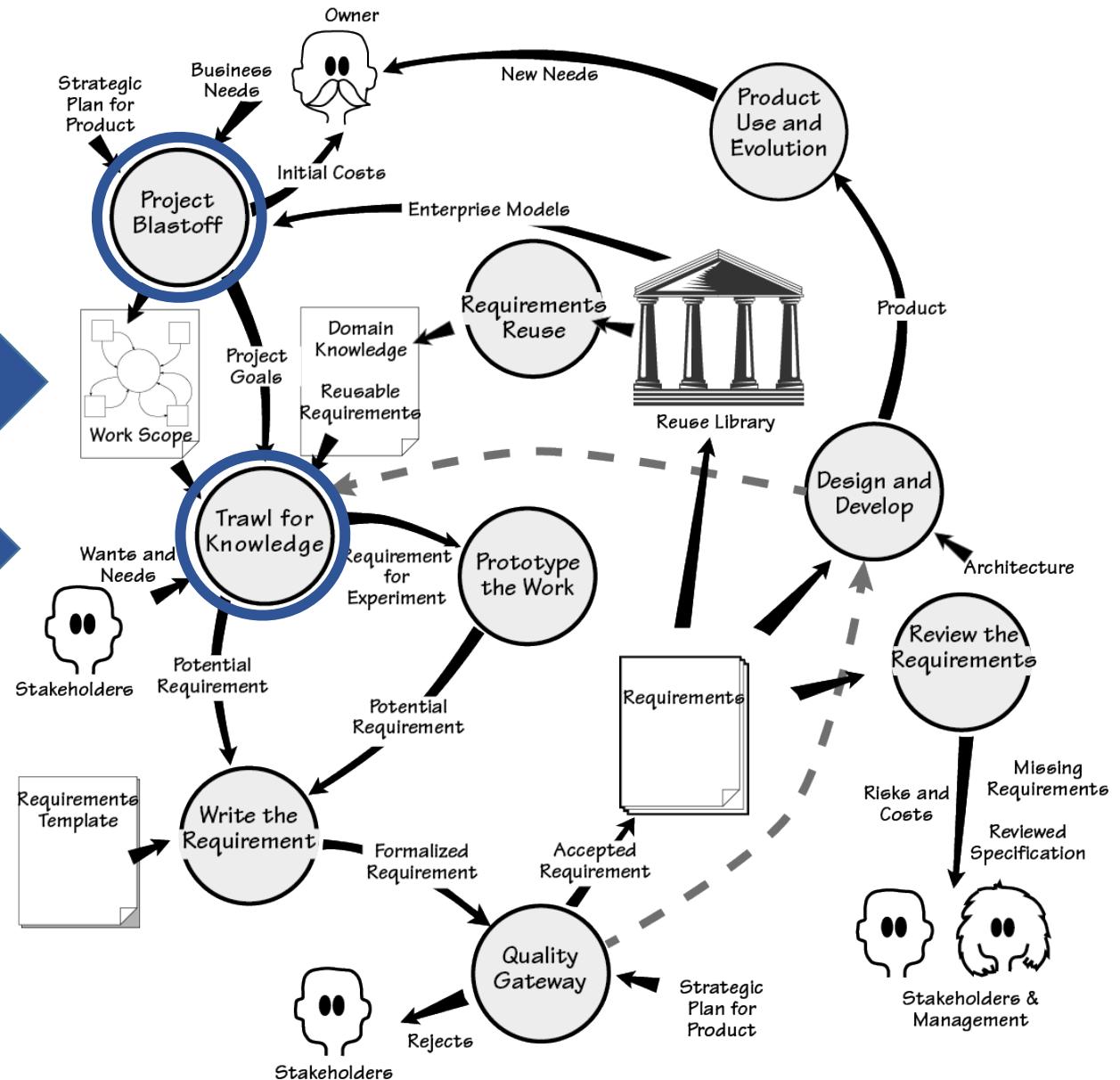
Project Issues

18. Open Issues
19. Off-the-Shelf Solutions
20. New Problems
21. Tasks
22. Migration to the New Product
23. Risks
24. Costs
25. User Documentation and Training
26. Waiting Room
27. Ideas for Solutions

Volere Requirements Process (Robertson)

Model system functionality

Concept of Operations



Project Blastoff (kickoff)

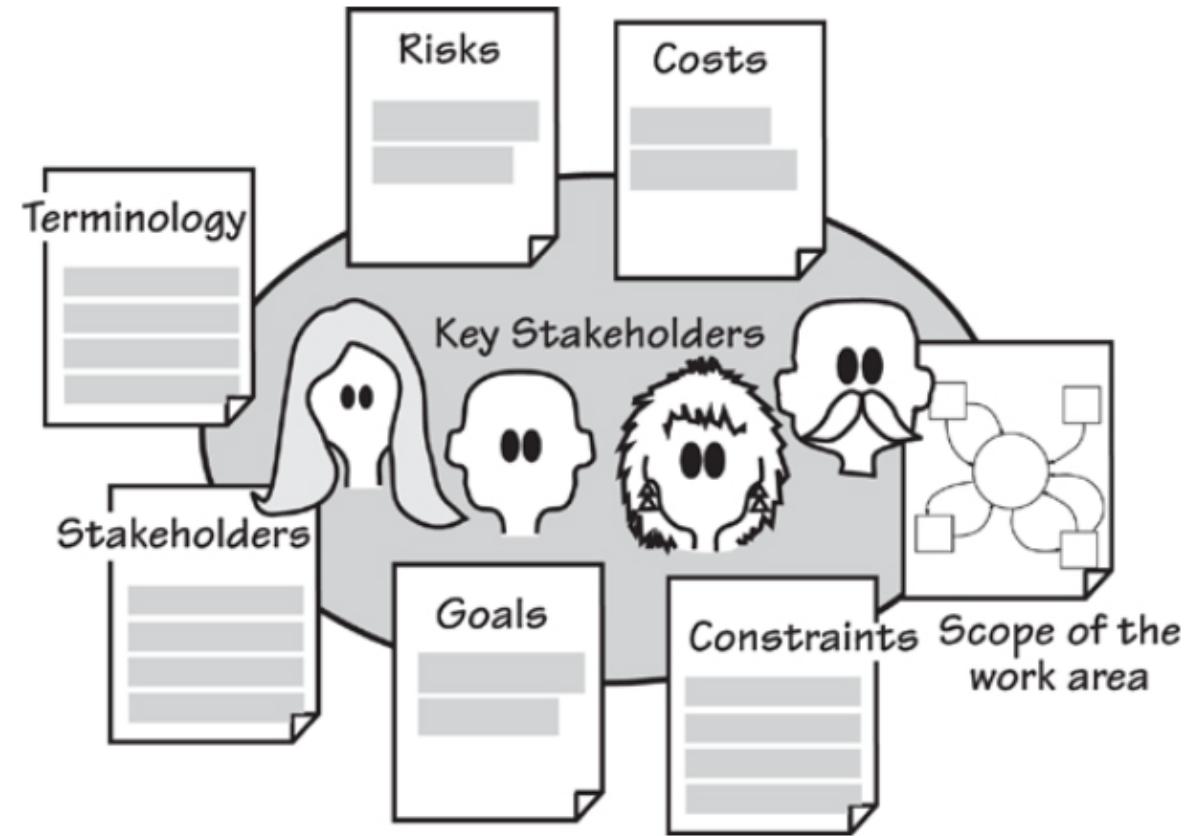
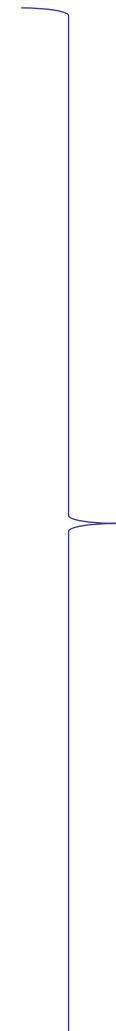
- The key purpose of project blastoff is to build the foundation for the requirements discovery, and to ensure that all the needed components for a **successful project** are in place.

- Involvement:

- Principal Stakeholder (sponsor)
- Key users
- Lead requirement analyst
- Technical business experts

- ▶ Discussions:

- Purpose of the project
- Scope of the work
- Stakeholders
- Constraints
- Included/excluded functionalities
- Special terminology
- Facts & Assumptions
- Cost estimates
- Risks



LECTURE 4 Continued

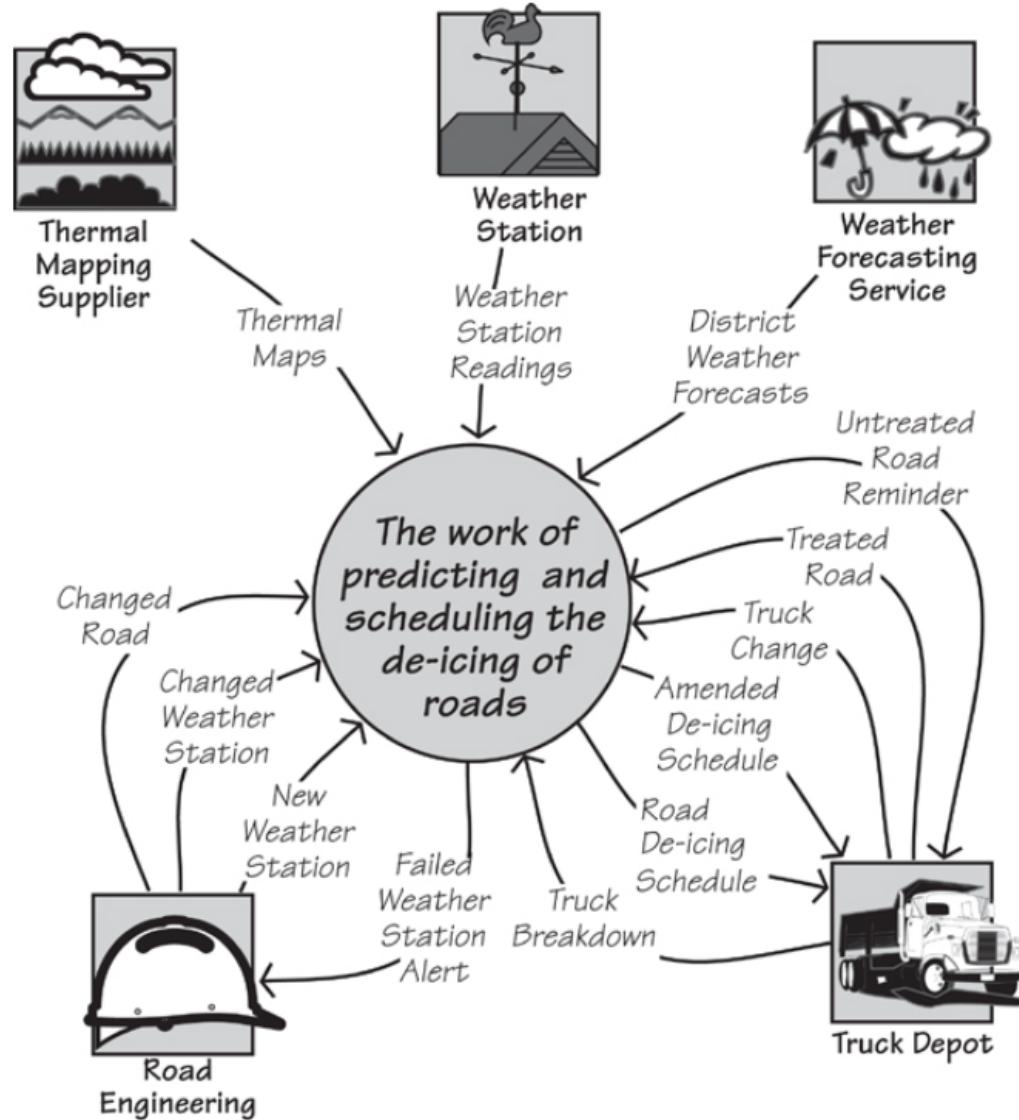
Importance of Product Scope

- A system react to external events
- Boundaries are not always well defined (soft systems, vs hard systems)
- Without boundaries scope creep is likely to occur
- Take time up front early in the project to define the scope of the product
- Understand the product context and interfaces with enough detail to be able to make accurate scoping decisions
- For some projects or program this may influence parts of the work and requirements that are “flown down” to external entities

Scoping the system allows separating the **Work** from its **Environment**

To achieve the optimal value for the owner, study enough of the owner's work to identify what is valuable (Robertson)

Context Diagram (Robertson)



The work context shows where the responsibilities of the work and the responsibilities of the adjacent systems start and end.

the responsibilities are defined by the flows of data on the context diagram. Always ask ...Why is this flow there?

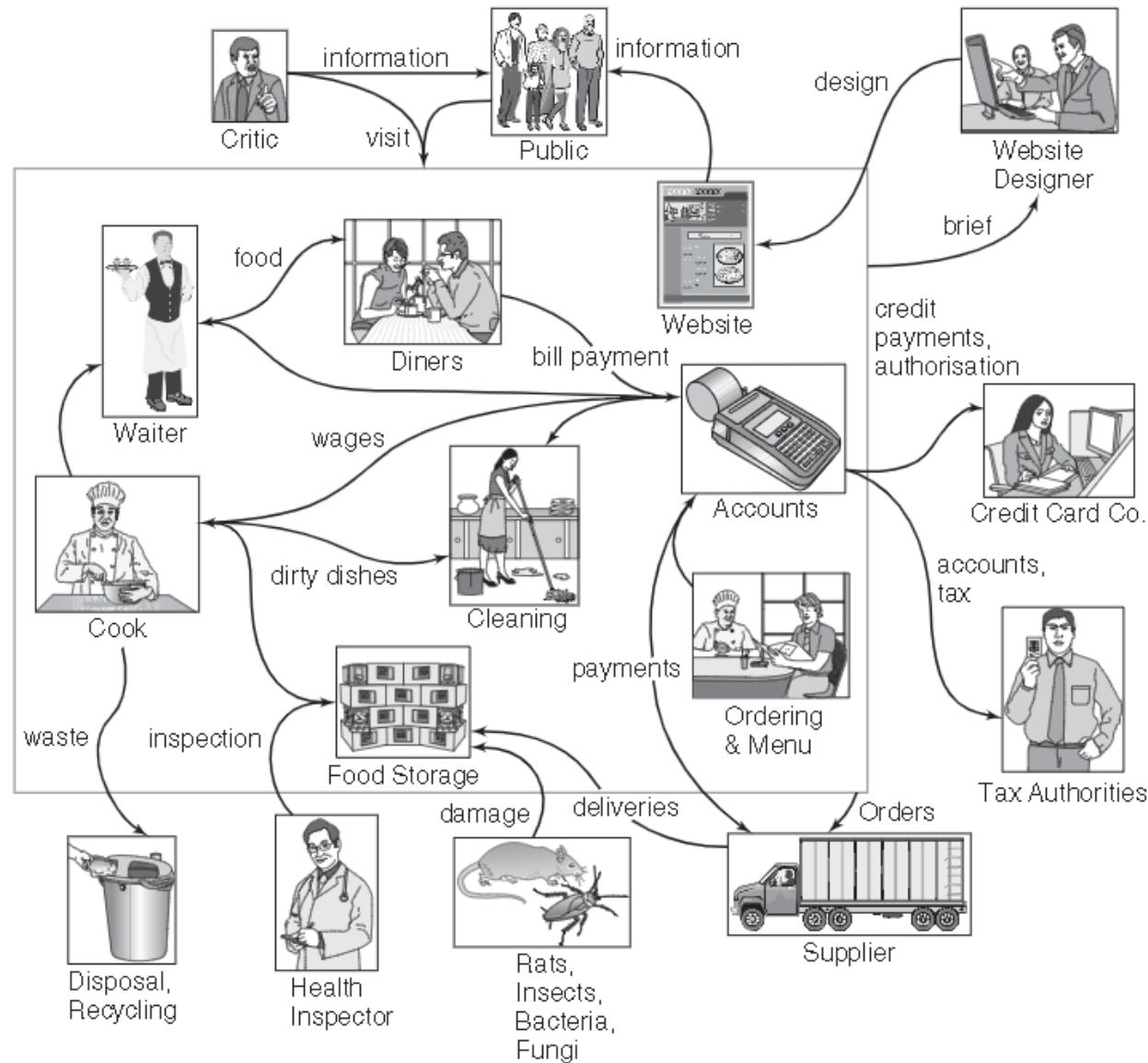
The context diagram intentionally ignores anything that happens outside the boundary that does not directly involve an interface with the system in question

Move from an soft system problem to a hard system problem

Soft System

- A soft system involves social, political and emotional issues as well as technology.
- Soft systems address messy , ill-structured problem situations
- A hard system may be in place , but it may fail for soft reasons:
 - IT solution to provide support, may fail because too complicated for some users
- Once implemented a product or service become part of the system
- Narrow system boundaries-> hard system
- Wide boundaries->soft system
- Draw a rich picture of the system showing:
 - Stakeholder not directly involved in the operations
 - Issues and concerns
 - Processes of stakeholders
 - Negative aspects
- The system boundary is outside the product boundary

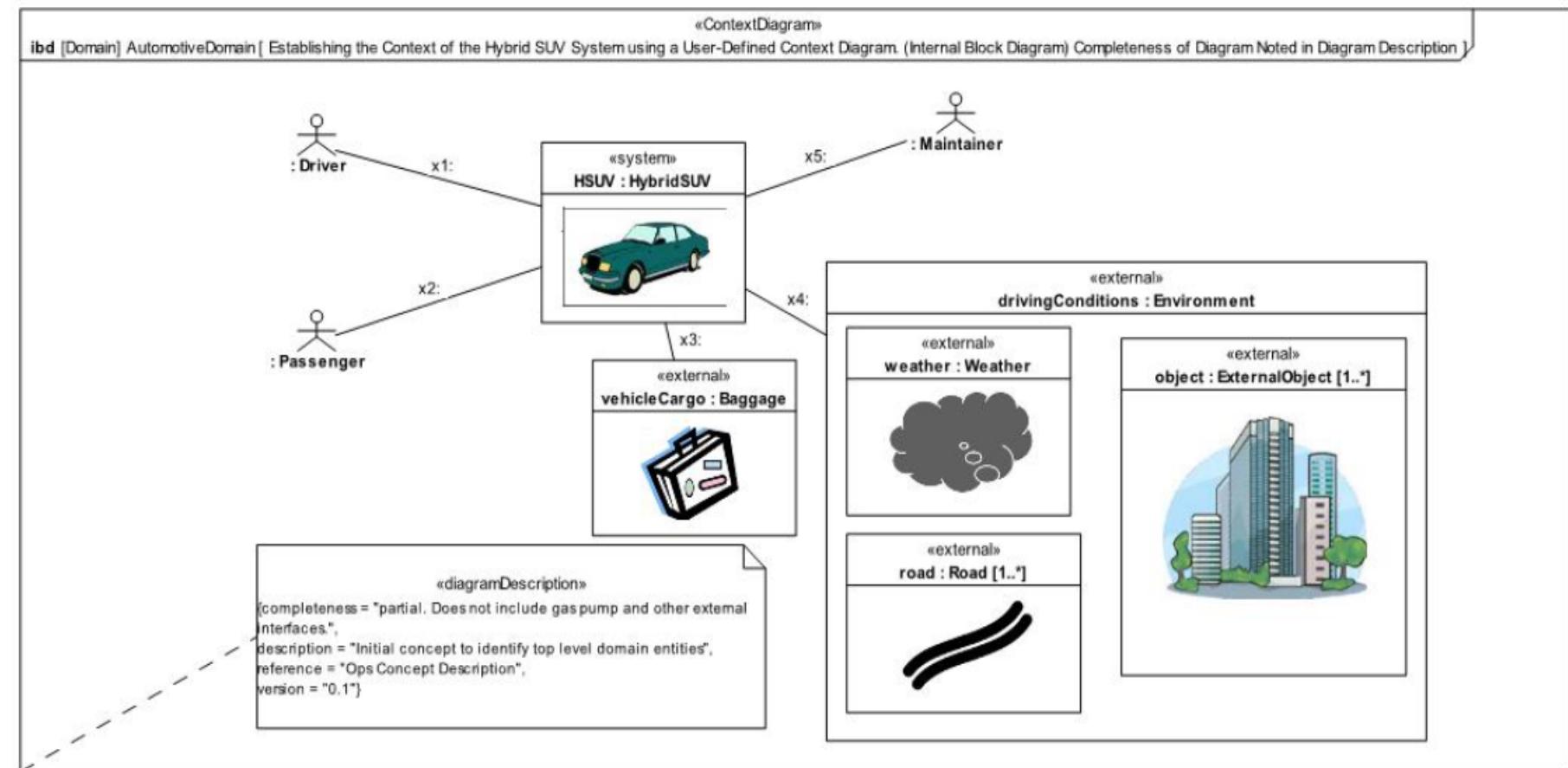
Example: Rich Picture of a Restaurant



What will be the system , if your product is the restaurant accounts?

Context Diagram of an SUV (SysML 1.6 Reference Document)

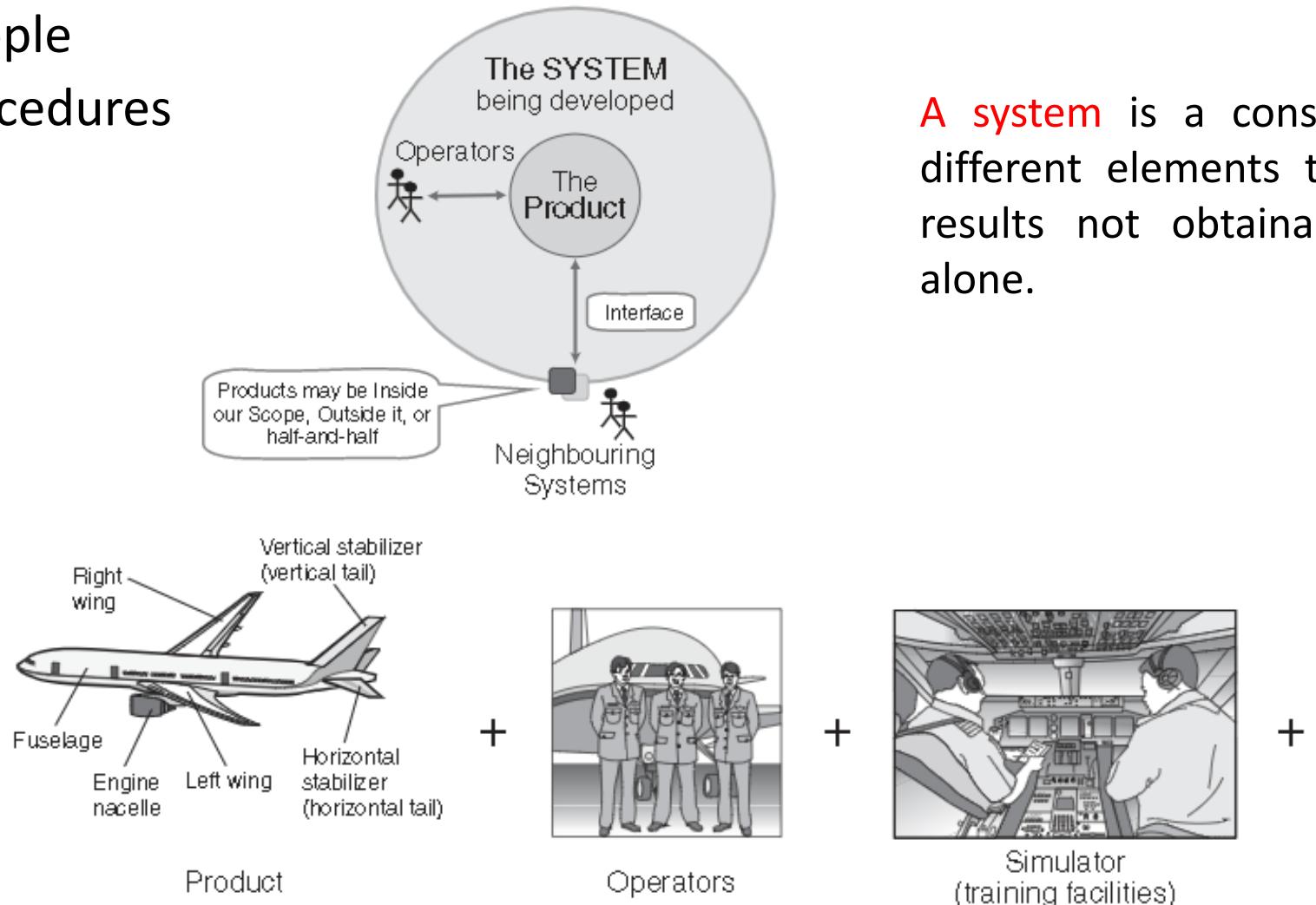
- An actor is an entity that interact with system
- Actors are outside the boundaries of the system
- Actors could be:
 - Hardware devices
 - Legacy systems
 - Other subsystems
 - Human users



System and its Context

- System consists on several components working together

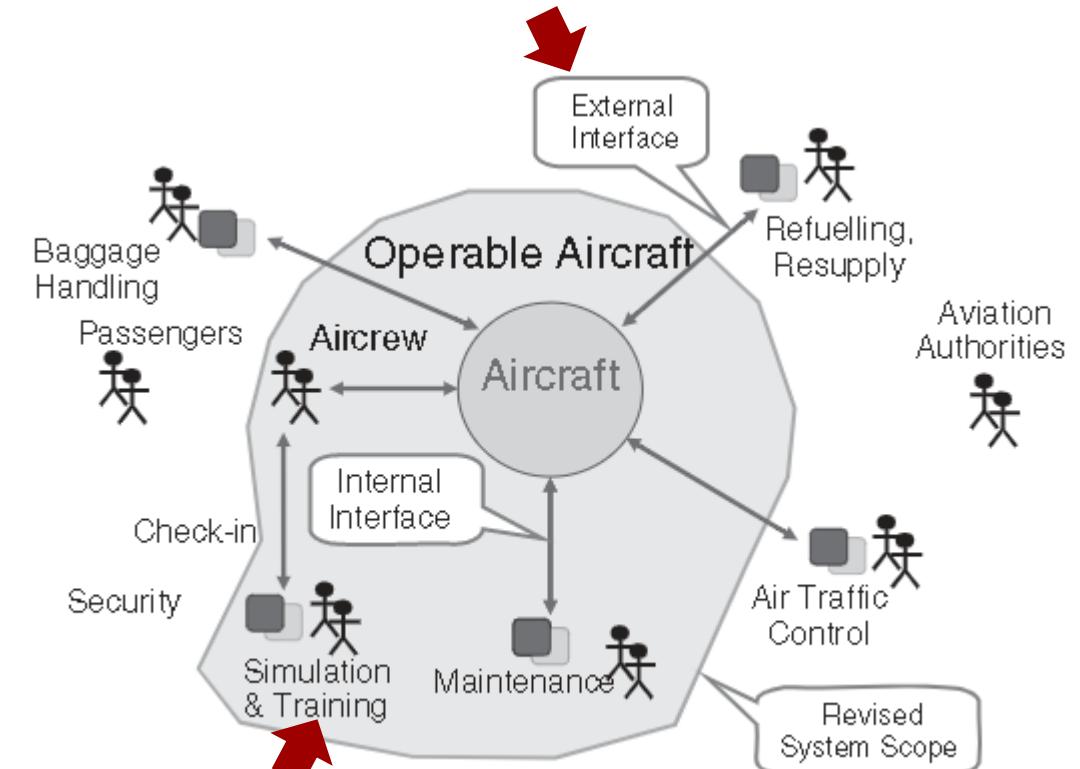
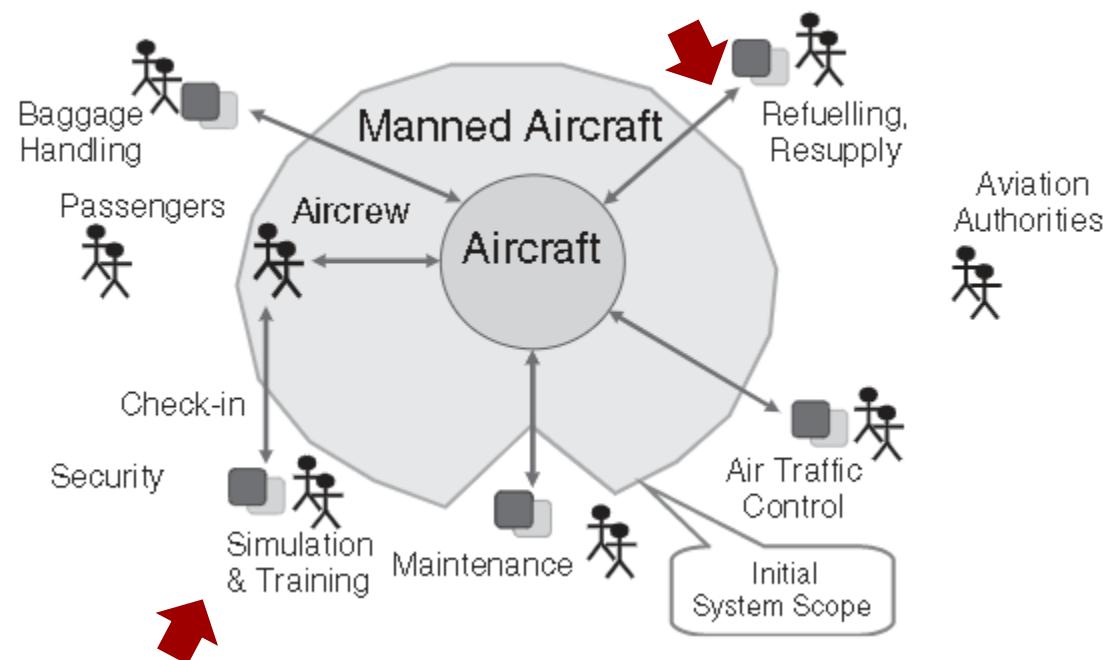
- Products
 - People
 - Procedures



A **system** is a construct or collection of different elements that together produce results not obtainable by the elements alone.

Example of Scope Definition

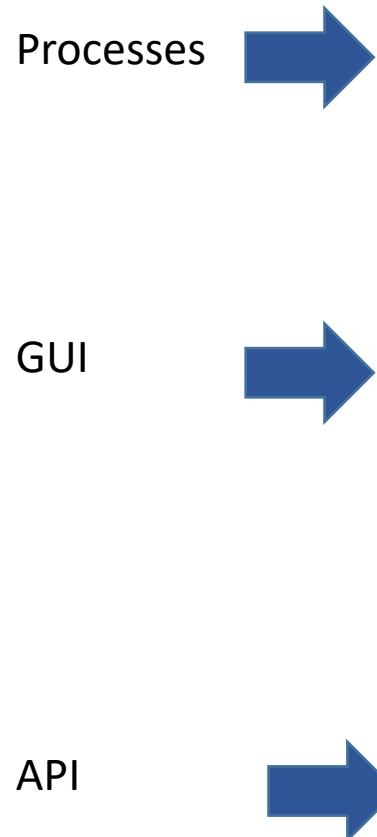
- Spot the differences in the two system representations
- Discuss on pros and cons of the two scopes



Identifying Interfaces

- Interface: a point where two systems, subjects, organizations, etc., meet and interact
- Interfaces are not necessarily hardware:
 - USB, Firewire
 - Gas tank and pump at gas stations
 - Mailbox
 - Point of contact
- Pay attention to interfaces among people (soft system)
- Create a context diagram

Interfaces and Requirements Elicitation Techniques

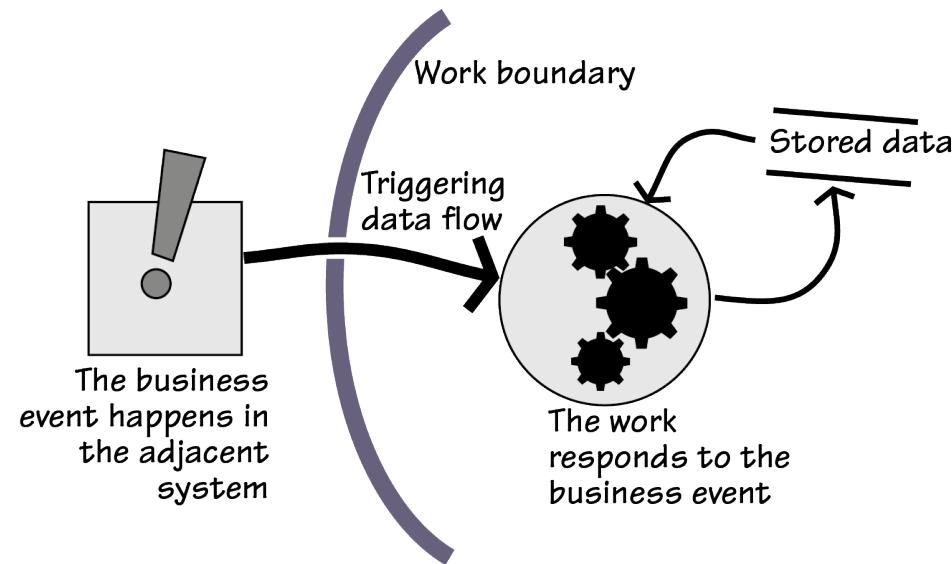


| Type of interface | Example | How to discover this | Treatment |
|--------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Between people | Aircrew report a problem to maintenance staff | Study the 'soft' system and its business processes (this chapter) Scenarios (Chapter 5) | Write a procedure; train crew to follow the procedure (i.e., write a training course, and run it) |
| Between people and products (These can be divided into inputs and outputs) | Aircrew interact with controls on the aircraft, both giving commands and receiving information | Scenarios (Chapter 5) Prototyping (Chapter 13) Similar/existing/rival products (Chapter 13) | Specify the user interface, considering effect on work (ergonomics); write a procedure for each use of the interface; train crew to follow the procedure (i.e., write a training course, and run it) |
| Between products (These can be in either direction, or both) | Automatic equipment diagnoses a fault on the aircraft | For interfaces to existing external systems: standards, or manufacturers' data sheets For interfaces to existing products/systems: speak to your developers/suppliers For new interfaces: iterate requirements and design (Chapter 14) | Specify the interface (the data and other quantities to be exchanged, and the required behaviour), using standards if possible; impose the interface as requirements on both products |

Scope and System Events

- Flows are provided as event that are enabled by the interfaces
- Draw or establish events on the boundary of the system
- An arrow into the circle on a context diagram means an event
 - Unpredictable arrival of a signal to which the **system has to respond by carrying out one or more tasks-> functional decomposition**
- It is a good practice to **start with a list of in/out events the system need to handle**
 - **(or the system needs to trigger, then handle...)**
- The scope consists on deciding the subset of events that will be handled by the system
- Handling such event become a requirement if agreed in the scope
- This has contractual implications for the product acceptance
- Events could be:
 - External: message, signal , control input
 - Time-triggered: time signal, polling cycle, etc.

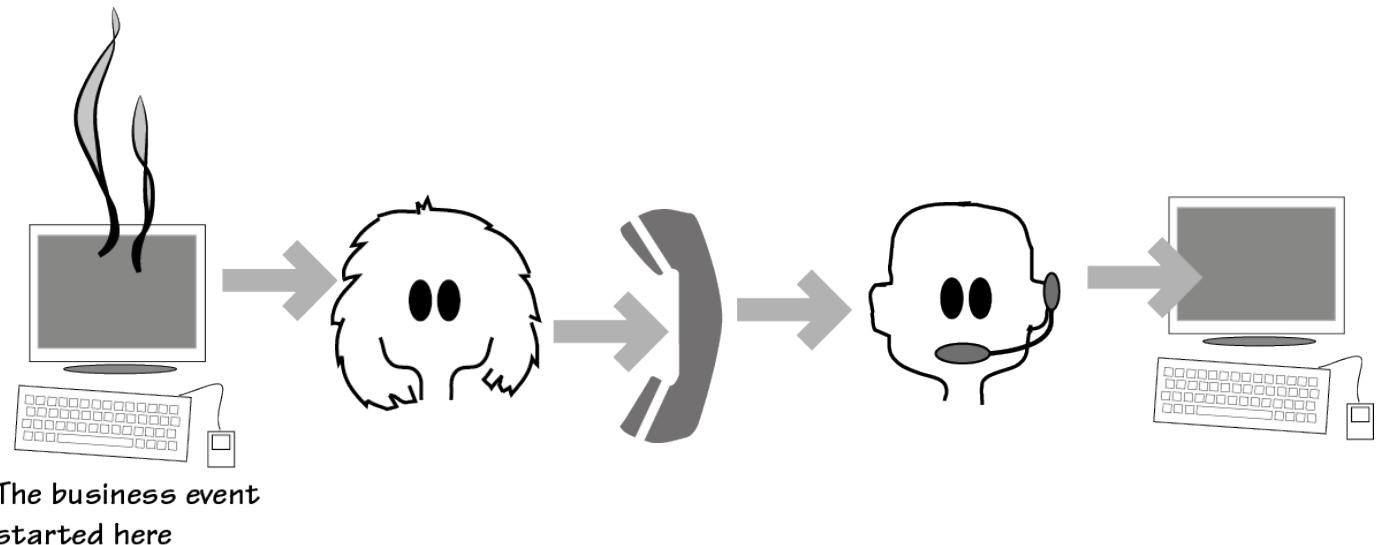
Flows are Triggered by Business Events (Ch4 Robertson)



Understanding of the sources of the business event allow to a better understanding of the system and to come up with innovative designs

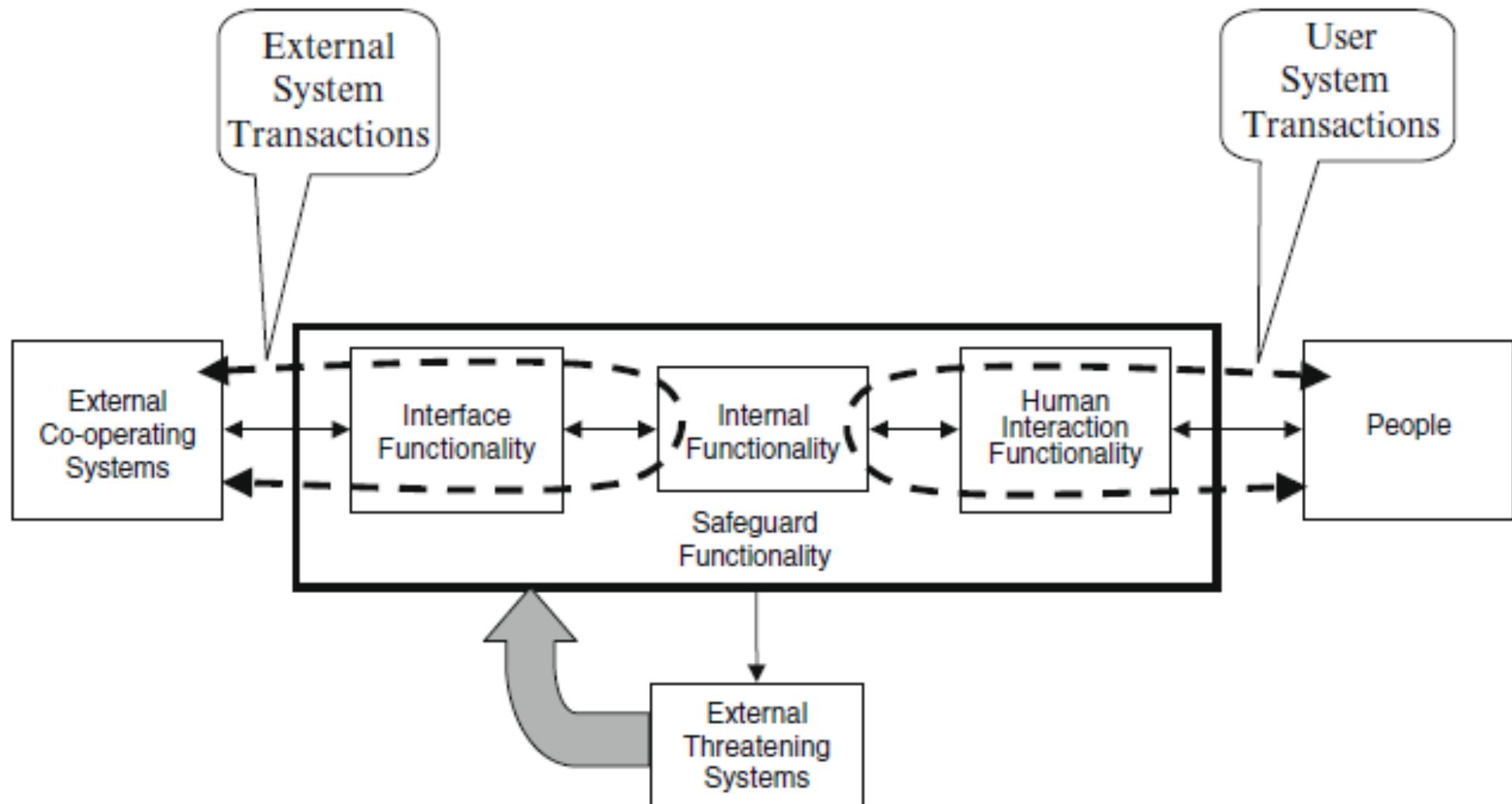
Business events can be time-based triggered (timers, or a monthly bill)

Establish the true actors of the system and flows by identifying business events



The maintenance call is received by the operator and input in the system *

Grouping Events Based on Interfaces (Hull, 2011)



Context Diagram and Events



| Event/Task | In/Out |
|---------------------------|--------------|
| Load Bags | In Scope |
| Unload Bags | In Scope |
| Refuel | In Scope |
| Check-in Passengers | Out of Scope |
| Radio Air Traffic Control | In Scope |
| Provide Airport Security | Out of Scope |
| | |
| | |

Event Handling Functions

- Express event handling by:
 - Textual event handling requirements
 - Condition-action tables
- Template for a traditional event handling requirement:
 - When<event is detected> do <required action>
 - Every <time period> do <required action>

When pressure in Main_Input_Pipe falls below Minimum_Input_Pressure, run Main_Pump at Full_Power.

Every Sample_Period, send Cylinder_Temperature to Engine_Controller.

When Payment_Confirmation is received, set Order_Status to To_Be_Despatched.

If Account_Balance exceeds Private_Banking_Threshold for Qualifying_Period then issue Private_Banking_Invitation to Customer.

Event Handling Functions(2)

■ Condition-Action Tables

- Condition action tables allow a more formal definition
- Conditions are expressed in columns (AND)
- Implication: conditions are mutually exclusive

| When condition₁ | and condition₂ | then |
|-----------------------------------------------------------------|----------------------------------|------------------------------------|
| <i>Pressure (Main_Input_Pipe) < Minimum_Input_Pressure</i> | | <i>Run Main_Pump at Full_Power</i> |
| <i>Pressure (Main_Input_Pipe) > = Minimum_Input_Pressure</i> | | <i>Main_Pump Inactive</i> |

Users and Scenarios

- Using the events and interfaces identified, lets generate user cases and system scenarios

As a <user role> when <event>, I want to be able to <capability> so that I can <goal>

| Requirement Elements | | Priorities |
|-------------------------|--|---------------------------|
| | | Measurements |
| | | Definitions |
| Discovery Contexts | | Rationale and Assumptions |
| Introduction | | Qualities and Constraints |
| From Individuals | | Scenarios |
| From Groups | | |
| From Things | | |
| Trade-Offs | | |
| Putting it all Together | | |

Context, Interfaces, Scope

- Interfaces=>enable events
- Scope=collection of events
- System capabilities allow stakeholder to achieve goals



How do we achieve capabilities?

Grouping Actions into Scenarios

- The system has to take several actions to achieve a capability
- This lead to the idea of scenario
- A scenario is a sequence of actions

System Output

- The system under design generates events that affect the world outside its boundaries
- Examples include:
 - Sending a message , sounding an alarm, etc.
- For a function or capability to be provided to a human operator:
 - The <operator> shall be enabled to< do the required action>
- For a product function:
 - The <product> shall <do the required actions> <with abc performance>

From Context to Use Case Diagrams

- Context diagram are not exactly use cases
- Context diagrams show interfaces not functions
- Use cases lists all the high-level functions that will allow the actors to achieve their goals (through the use of the system of course)

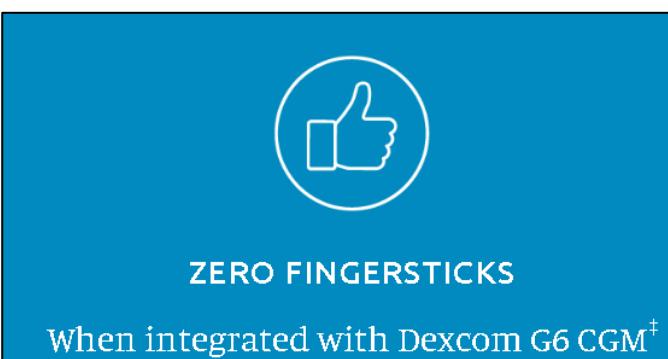
Sometimes the system is in the interface

dexcom
CONTINUOUS GLUCOSE MONITORING



TANDEM®
DIABETES CARE

t:slim X2 Insulin Pump
Easy to use. Easy to love.

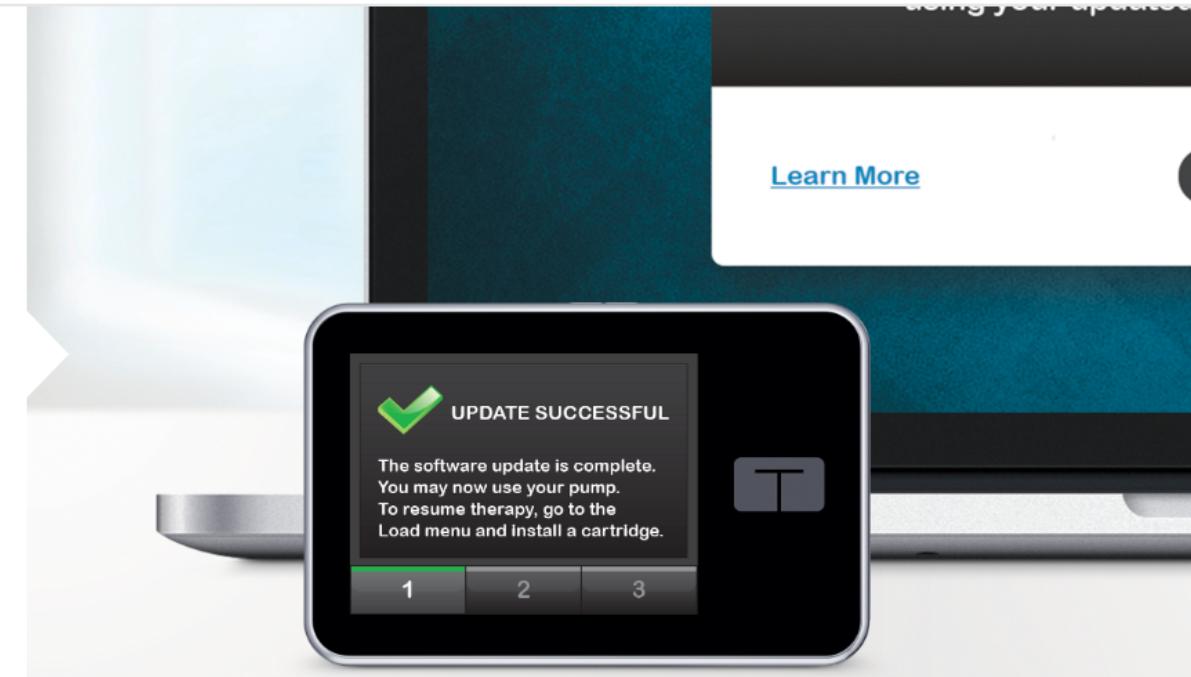


- Requirements exist for the lifecycle of the product

[PRODUCTS](#)[SUPPORT](#)[ABOUT](#)[CONTACT](#)[GET STARTED](#)

Remote software updates

The t:slim X2 insulin pump is the first FDA-approved insulin pump capable of remote feature updates.[†] Using a personal computer, patients can keep their pump up to date with the latest technology during its warranty period.



You know interfaces are important when...

Amazon, Apple, and Google unveil smart home collaboration



By [Pablo Valerio](#)

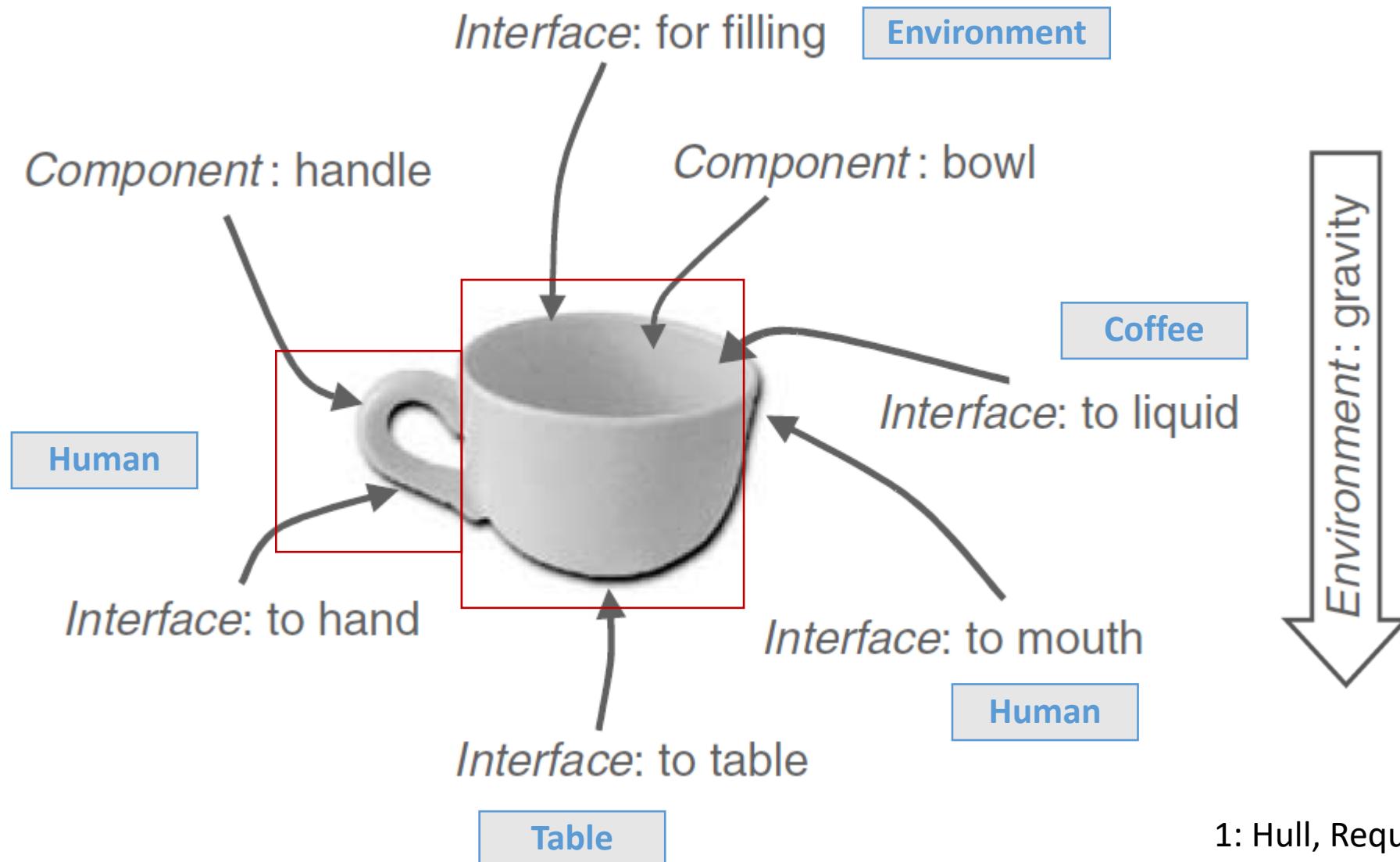
Posted on December 31, 2019



Interfaces and N-Squared Diagram

- The N-squared (N^2) diagram is used to determine system interfaces between components or functions.
- Rules:
 - The system components or functions are placed on the diagonal
 - Off-diagonal elements represent the interface inputs and outputs
 - Outputs are represented by rows
 - Inputs are represented by columns
 - Blanks mean no interface between the respective i-j components or functions
- Other usage
 - pinpoints areas where conflicts could arise in interfaces, and
 - highlights input and output dependency assumptions
- Not a formal SysML diagram

A Cup Viewed as a System¹



1: Hull, Requirements Engineering

- **Interface Requirements Document (IRD):** Defines the functional, performance, electrical, environmental, human, and physical requirements and constraints that exist at a common boundary between two or more functions or system elements
- **Interface Control Document or Interface Control Drawing (ICD):** Details the physical interface between two system elements, including the number and types of connectors, electrical parameters, mechanical properties, and environmental constraints.
- **Interface Definition Document (IDD) :** A unilateral document controlled by the end item provider. Provides the details of the interface for a design solution that is already established.

Validate Interfaces and Events

- For standard interfaces, ensure that the exact version of the relevant standard is identified
- For custom interfaces, agree all details of each interface with the owner of the other system
- Each event should be handled by when-requirements or their equivalent condition-action tables
- List of alternative conditions in condition-action tables cover all the possibilities exhaustively
- Check that each scenario referenced in the event handling requirement is defined as a use case
- Check that each interface is defined in the data dictionary

Group Activity

Get in new groups of 4



WHEREWELEARN

Demo Button Demo Button

Demo Button Demo Button Demo Button

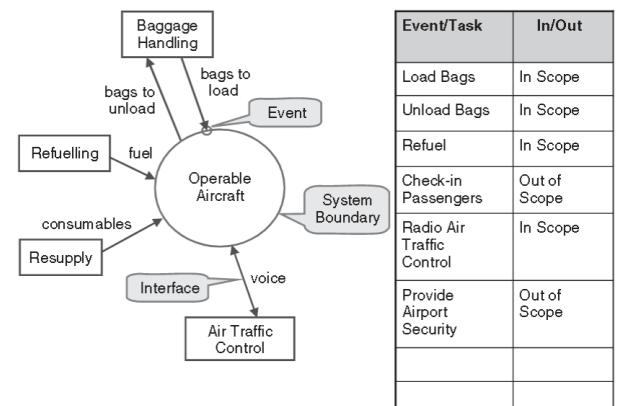
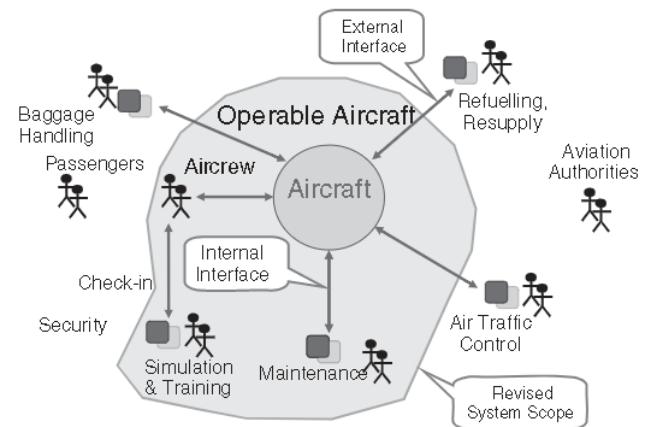
Demo Button Demo Button

MASTER SYSTEMS DISPLAY



In Class Exercise

- Create a Context Diagram and Events for the selected system.
- Include as many interfaces and events as you can identify
- Present to the class in under 3 min
- May use laptop or paper to present.



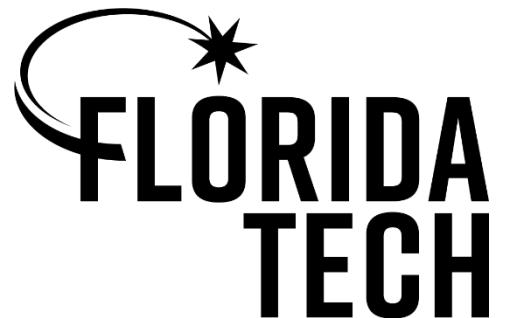
SYS 5460: Context , Interfaces, Scope

Contents

- Concept of Operations
- System Context
- Interfaces
- Examples

**Department of Computer &
Engineering Sciences**

College of Engineering
Florida Institute of Technology





How the customer
explained it



How the project leader
understood it



How the analyst
designed it



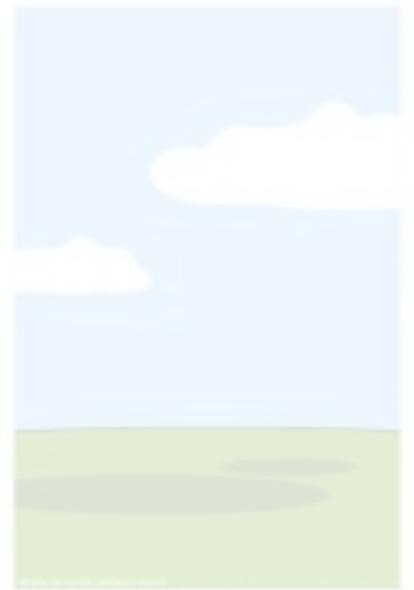
How the programmer
wrote it



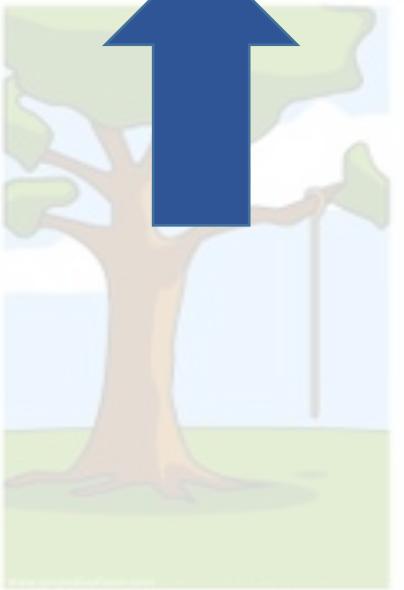
What the beta testers
received



How the business
consultant described it



How the project was
documented



What operations
installed



How the customer was
billed



How it was supported



iSwing

What marketing
advertised

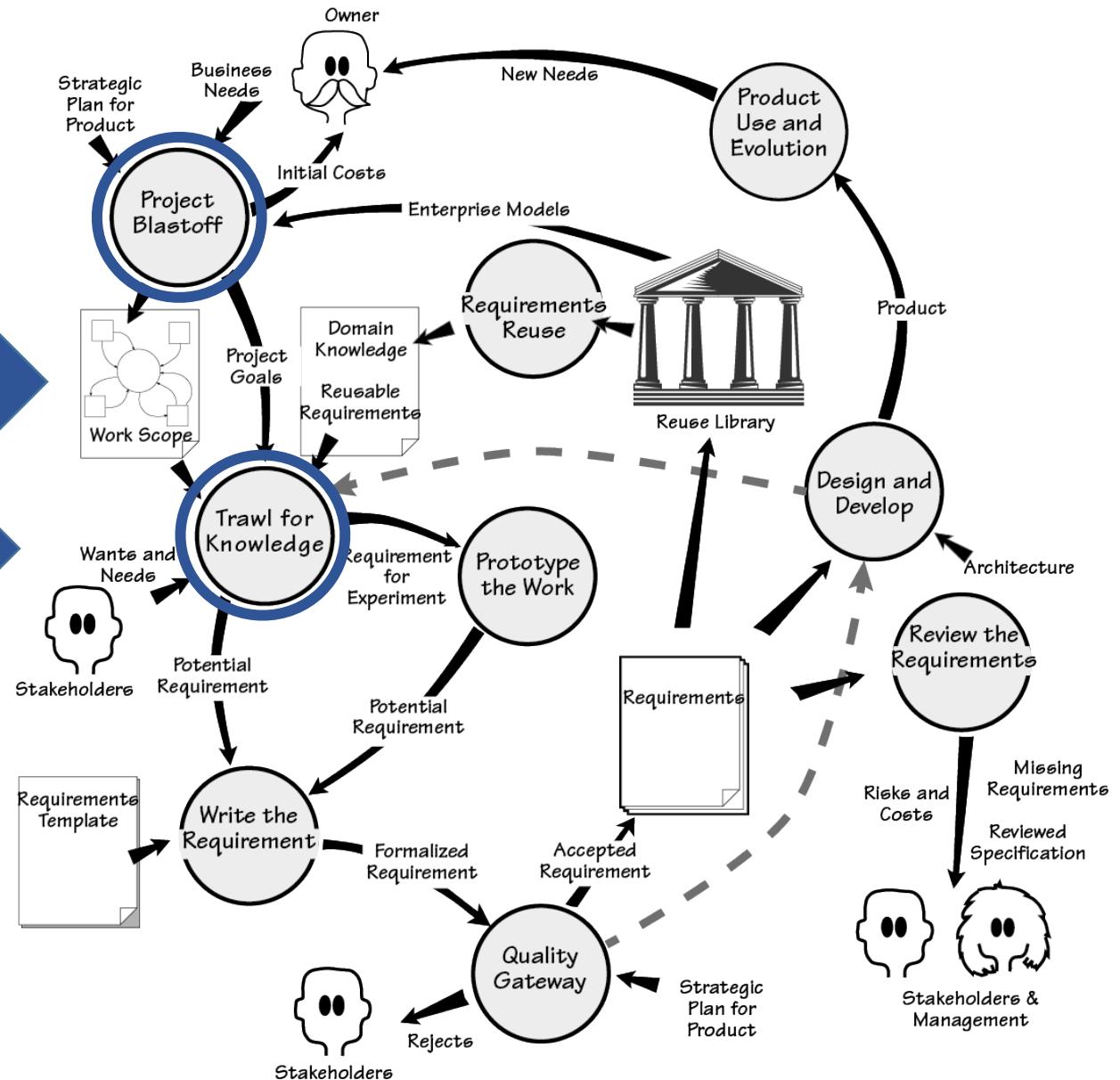


What the customer
really needed

Volere Requirements Process (Robertson)

Model system functionality

Concept of Operations



Standards For Concept of Operations

Previous standards

ANSI/AIAA-G-043 Outline

1. Scope
2. Referenced Documents
3. User-Oriented Operational Description
4. Operational Needs
5. System Overview
6. Operational Environment
7. Support Environment
8. Operational Scenarios

IEEE 1362 Outline

1. Scope
2. Referenced Documents
3. The Current System or Situation
4. Justification for and Nature of Changes
5. Concepts for the Proposed System
6. Operational Scenarios
7. Summary of Impacts
8. Analysis of the Proposed System



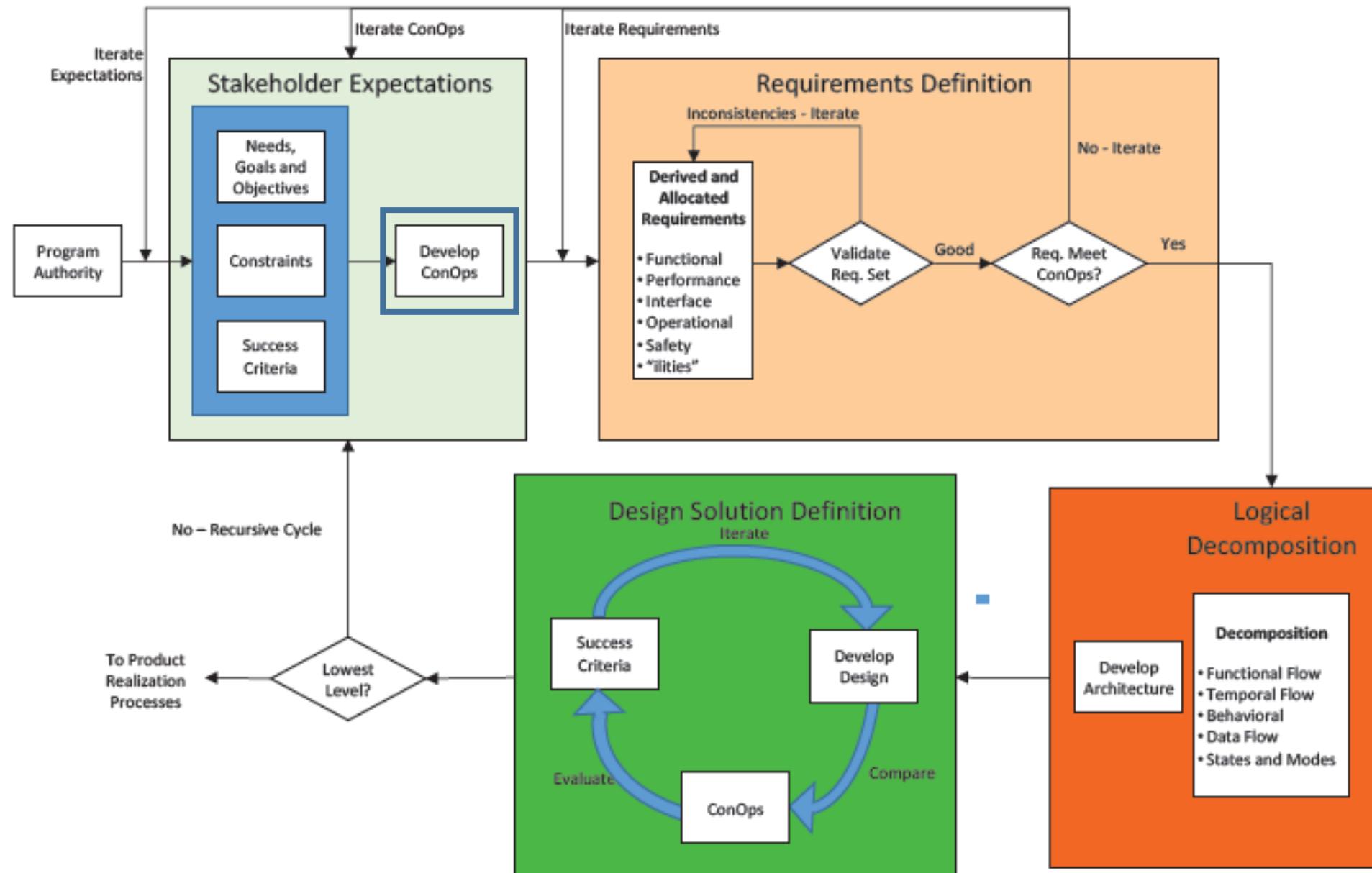
Current standards ISO/IEC 12207 Software lifecycle process or ISO/IEC 15288 Systems and Software lifecycle processes

concept of operations

verbal and/or graphic statement, in broad outline, of an organization's assumptions or intent in regard to an operation or series of operations

Note 1 to entry: The concept of operations frequently is embodied in long-range strategic plans and annual operational plans. In the latter case, the concept of operations in the plan covers a series of connected operations to be carried out simultaneously or in succession. The concept is designed to give an overall picture of the organization operations. See also operational concept.

Note 2 to entry: It provides the basis for bounding the operating space, system capabilities, interfaces and operating environment.



DoDAF Viewpoints and Models

OV-1: High-Level Operational Concept Graphic

OV-2: Operational Resource Flow Description

OV-3: Operational Resource Flow Matrix

OV-4: Organizational Relationships Chart

OV-5a: Operational Activity Decomposition Tree

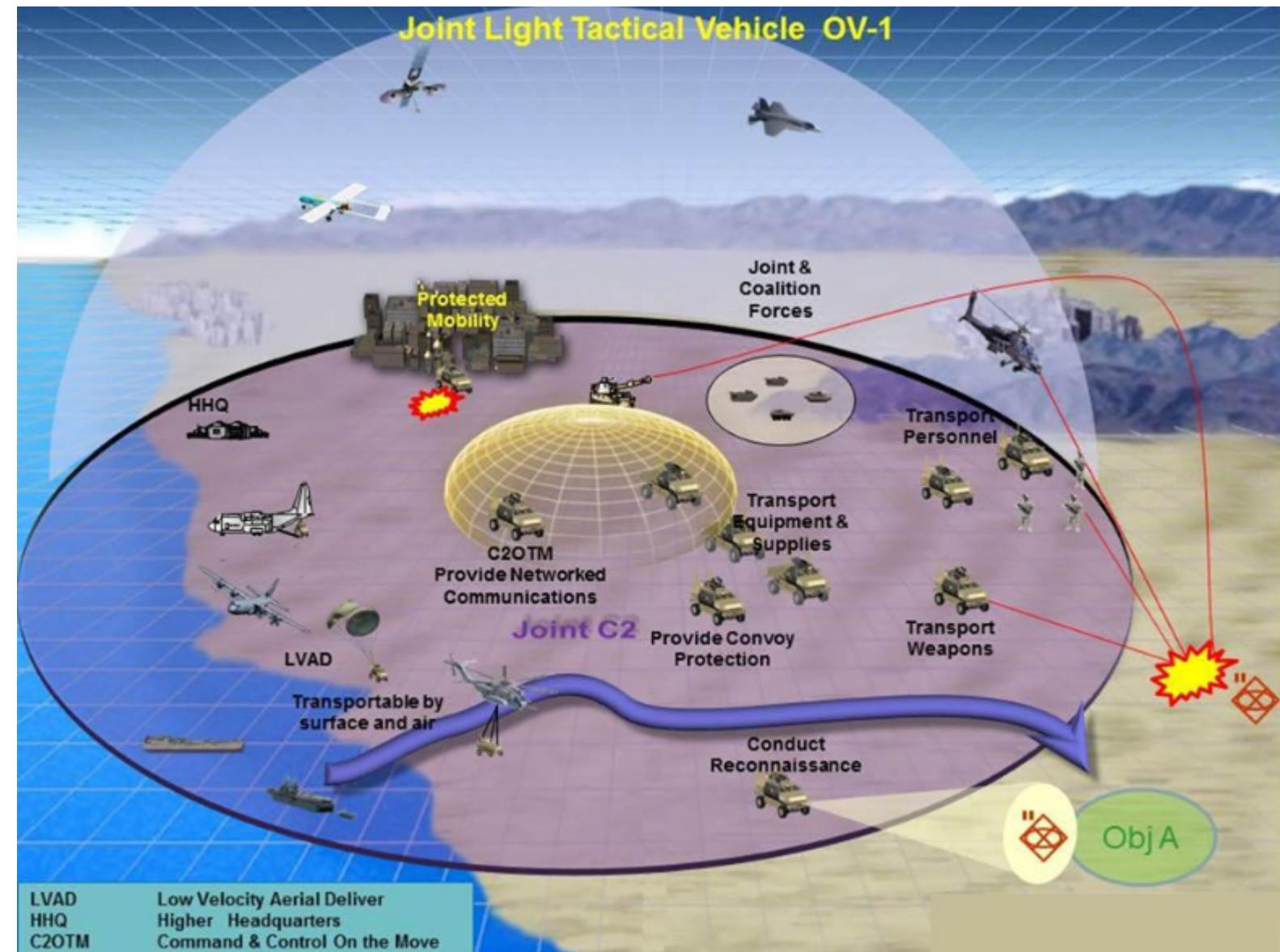
OV-5b: Operational Activity Model

OV-6a, 6b, 6c: Introduction

OV-6a: Operational Rules Model

OV-6b: State Transition Description

OV-6c: Event-Trace Description



Concept of Operations Document (IEEE 1362-1998)

Title page

Revision chart

Preface

Table of contents

List of figures

List of tables

1. Scope

 1.1 Identification

 1.2 Document overview

 1.3 System overview

2. Referenced documents

3. Current system or situation

 3.1 Background, objectives, and scope

 3.2 Operational policies and constraints

 3.3 Description of the current system or situation

 3.4 Modes of operation for the current system or situation

 3.5 User classes and other involved personnel

 3.6 Support environment

4. Justification for and nature of changes

 4.1 Justification of changes

 4.2 Description of desired changes

 4.3 Priorities among changes

 4.4 Changes considered but not included

5. Concepts for the proposed system

 5.1 Background, objectives, and scope

 5.2 Operational policies and constraints

 5.3 Description of the proposed system

 5.4 Modes of operation

 5.5 User classes and other involved personnel

 5.6 Support environment

6. Operational scenarios

7. Summary of impacts

 7.1 Operational impacts

 7.2 Organizational impacts

 7.3 Impacts during development

8. Analysis of the proposed system

 8.1 Summary of improvements

 8.2 Disadvantages and limitations

 8.3 Alternatives and trade-offs considered

9. Notes

Appendices

Glossary

ConOps

Project Drivers

1. The Purpose of the Project
2. The Stakeholders

Project Constraints

3. Mandated Constraints
4. Naming Conventions and Terminology
5. Relevant Facts and Assumptions

Functional Requirements

6. The Scope of the Work
7. The Business Data Model and Data Dictionary
8. The Scope of the Product
9. Functional Requirements

Non-functional Requirements

10. Look and Feel Requirements
11. Usability and Humanity Requirements
12. Performance Requirements
13. Operational and Environmental Requirements
14. Maintainability and Support Requirements
15. Security Requirements
16. Cultural Requirements
17. Legal Requirements

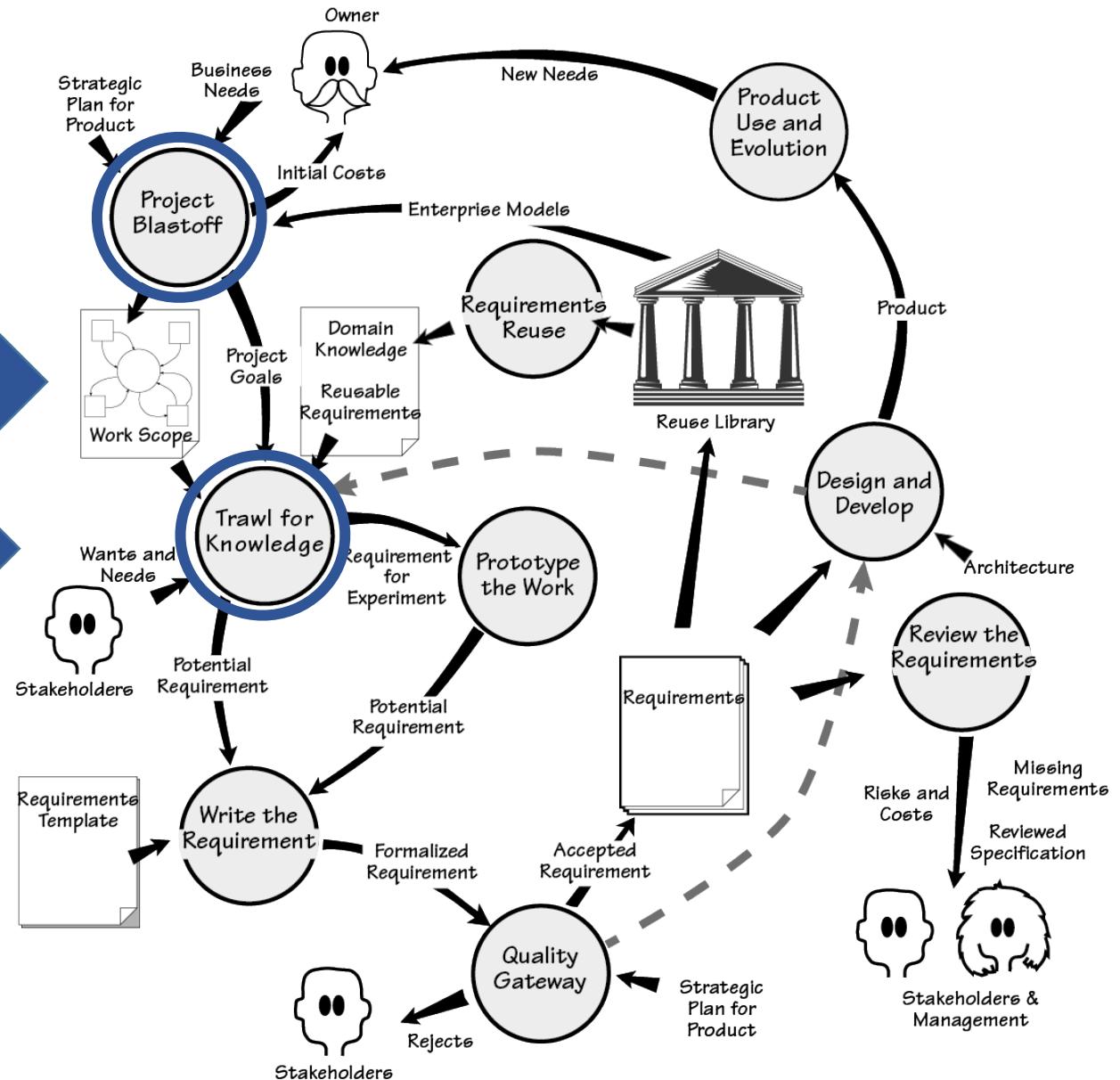
Project Issues

18. Open Issues
19. Off-the-Shelf Solutions
20. New Problems
21. Tasks
22. Migration to the New Product
23. Risks
24. Costs
25. User Documentation and Training
26. Waiting Room
27. Ideas for Solutions

Volere Requirements Process (Robertson)

Model system functionality

Concept of Operations



Project Blastoff (kickoff)

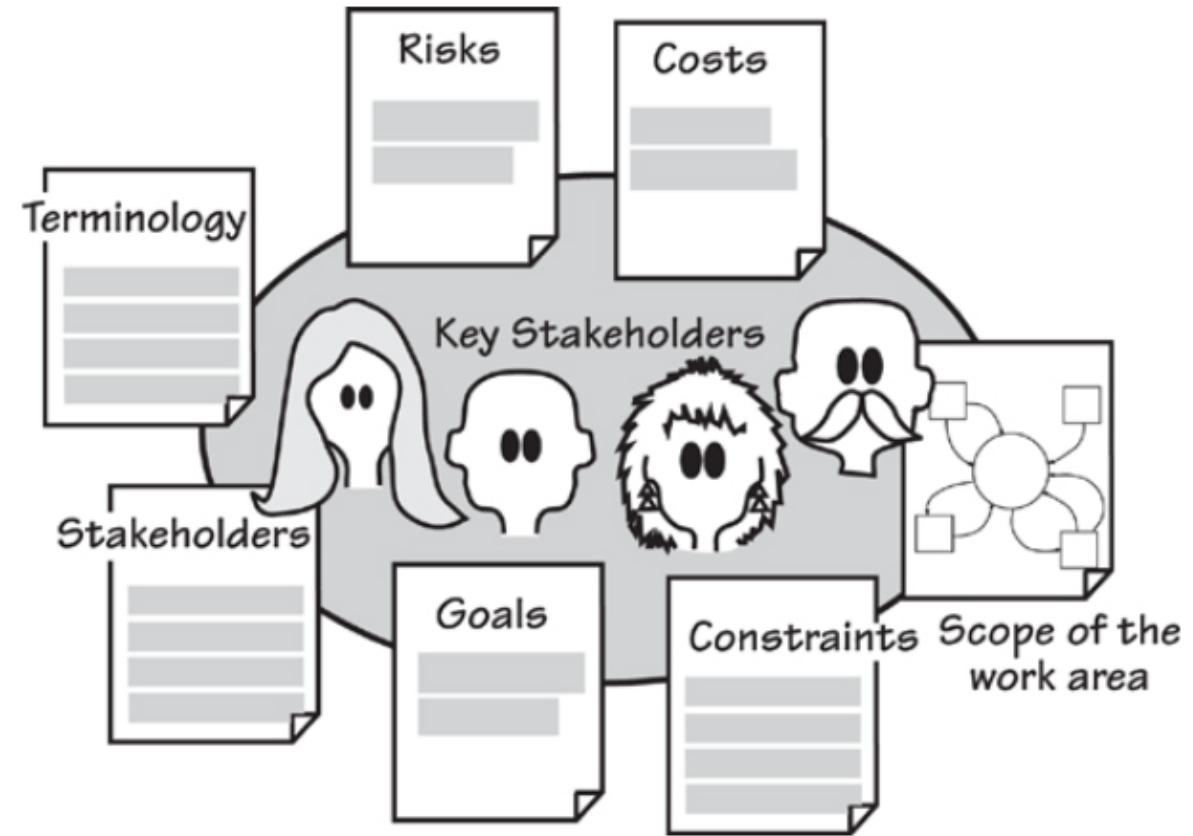
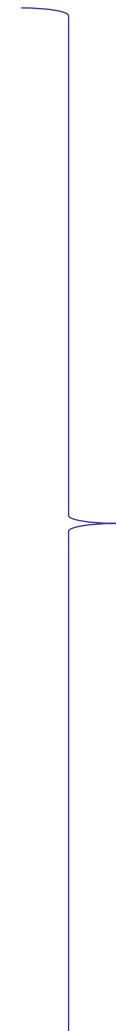
- The key purpose of project blastoff is to build the foundation for the requirements discovery, and to ensure that all the needed components for a **successful project** are in place.

- Involvement:

- Principal Stakeholder (sponsor)
- Key users
- Lead requirement analyst
- Technical business experts

- ▶ Discussions:

- Purpose of the project
- Scope of the work
- Stakeholders
- Constraints
- Included/excluded functionalities
- Special terminology
- Facts & Assumptions
- Cost estimates
- Risks



LECTURE 4 Continued

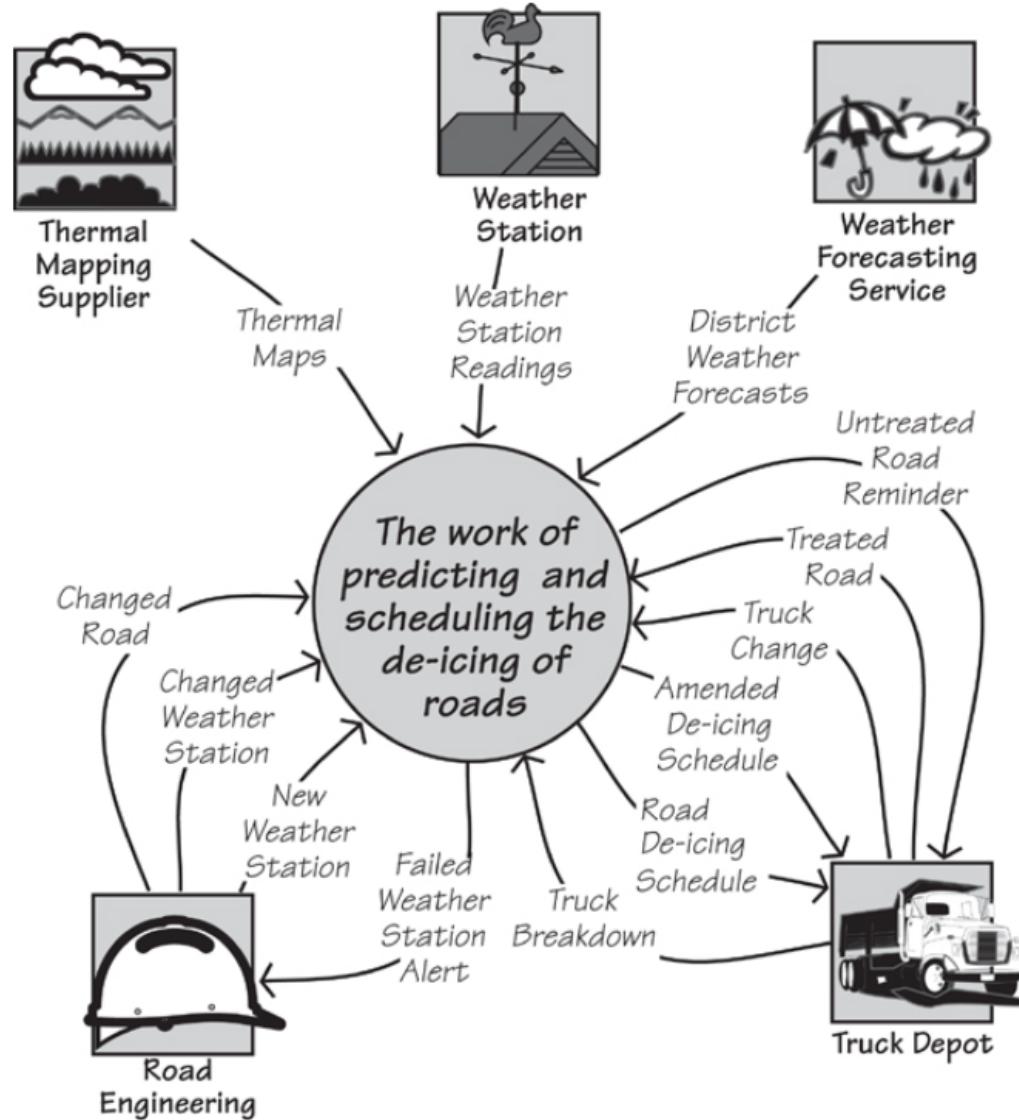
Importance of Product Scope

- A system react to external events
- Boundaries are not always well defined (soft systems, vs hard systems)
- Without boundaries scope creep is likely to occur
- Take time up front early in the project to define the scope of the product
- Understand the product context and interfaces with enough detail to be able to make accurate scoping decisions
- For some projects or program this may influence parts of the work and requirements that are “flown down” to external entities

Scoping the system allows separating the **Work** from its **Environment**

To achieve the optimal value for the owner, study enough of the owner's work to identify what is valuable (Robertson)

Context Diagram (Robertson)



The work context shows where the responsibilities of the work and the responsibilities of the adjacent systems start and end.

the responsibilities are defined by the flows of data on the context diagram. Always ask ...Why is this flow there?

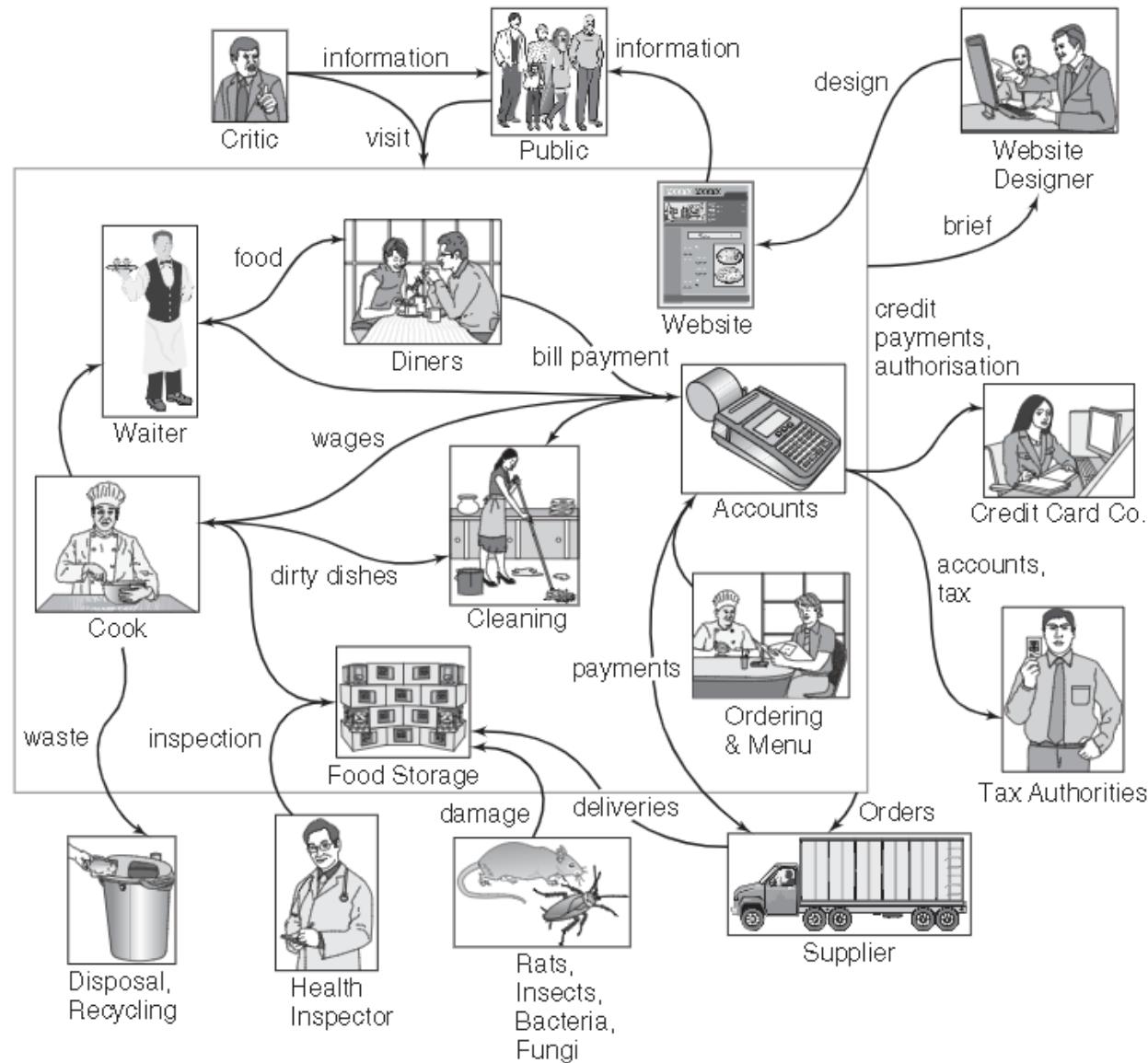
The context diagram intentionally ignores anything that happens outside the boundary that does not directly involve an interface with the system in question

Move from an soft system problem to a hard system problem

Soft System

- A soft system involves social, political and emotional issues as well as technology.
- Soft systems address messy , ill-structured problem situations
- A hard system may be in place , but it may fail for soft reasons:
 - IT solution to provide support, may fail because too complicated for some users
- Once implemented a product or service become part of the system
- Narrow system boundaries-> hard system
- Wide boundaries->soft system
- Draw a rich picture of the system showing:
 - Stakeholder not directly involved in the operations
 - Issues and concerns
 - Processes of stakeholders
 - Negative aspects
- The system boundary is outside the product boundary

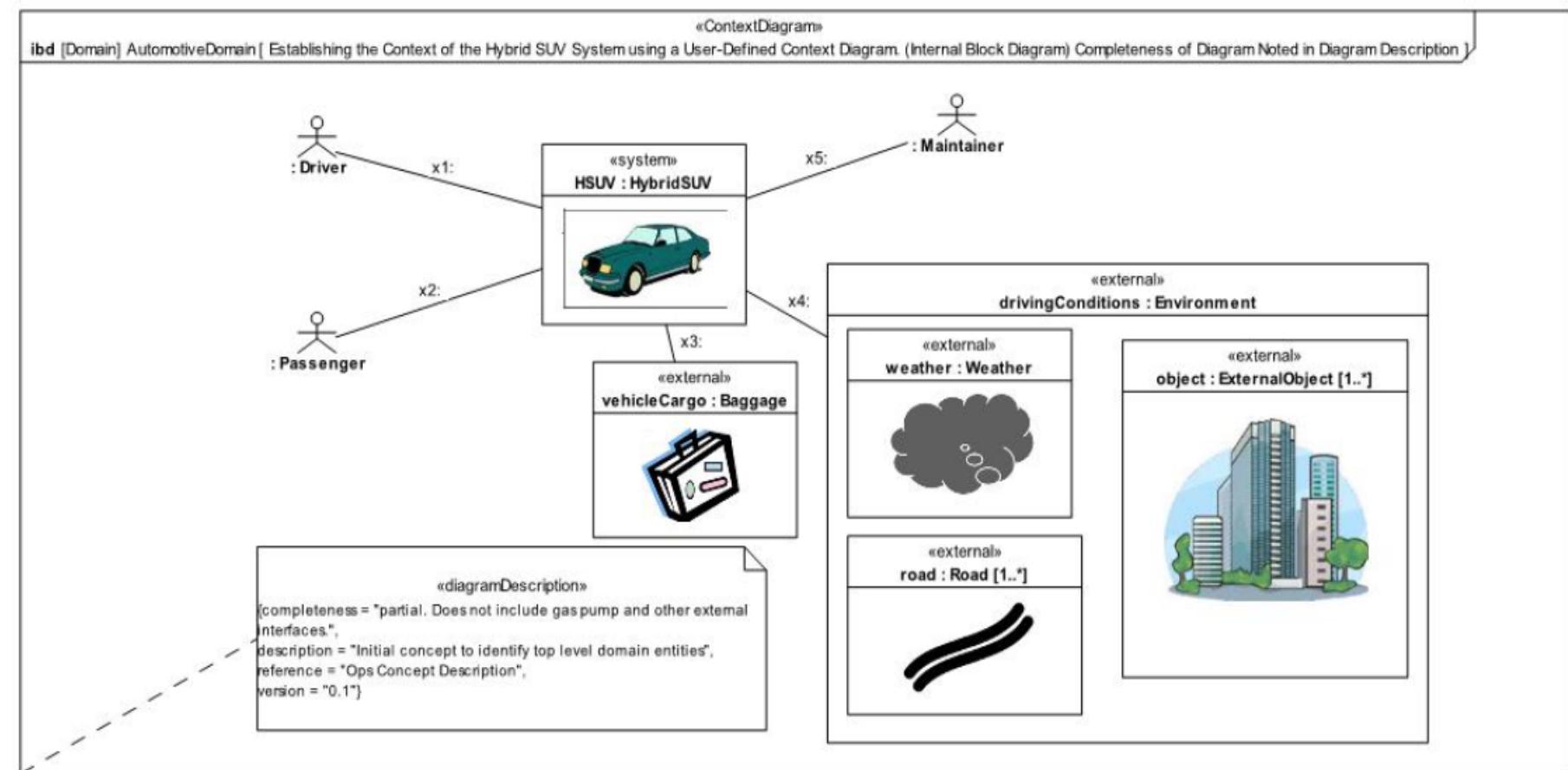
Example: Rich Picture of a Restaurant



What will be the system , if your product is the restaurant accounts?

Context Diagram of an SUV (SysML 1.6 Reference Document)

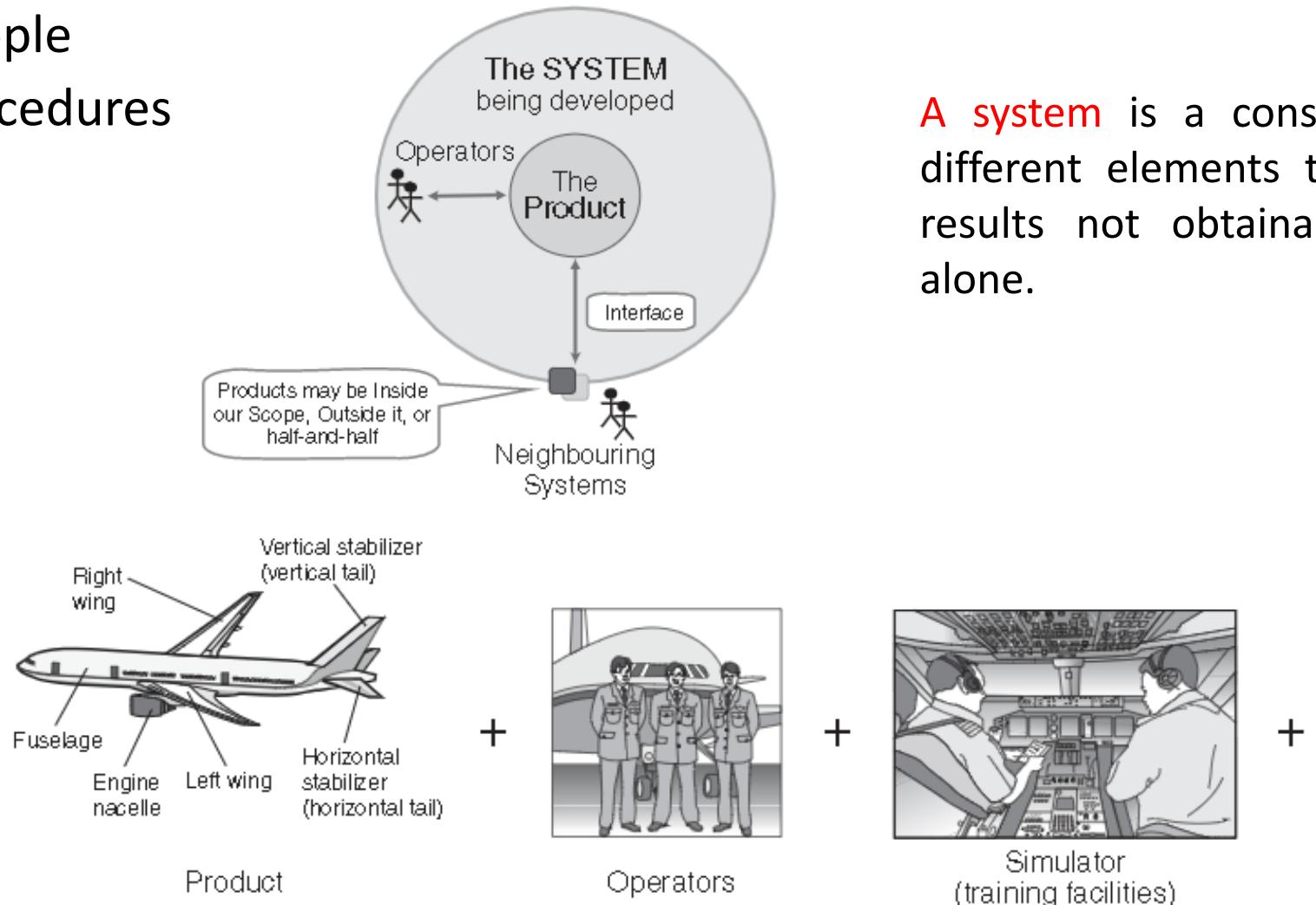
- An actor is an entity that interact with system
- Actors are outside the boundaries of the system
- Actors could be:
 - Hardware devices
 - Legacy systems
 - Other subsystems
 - Human users



System and its Context

- System consists on several components working together

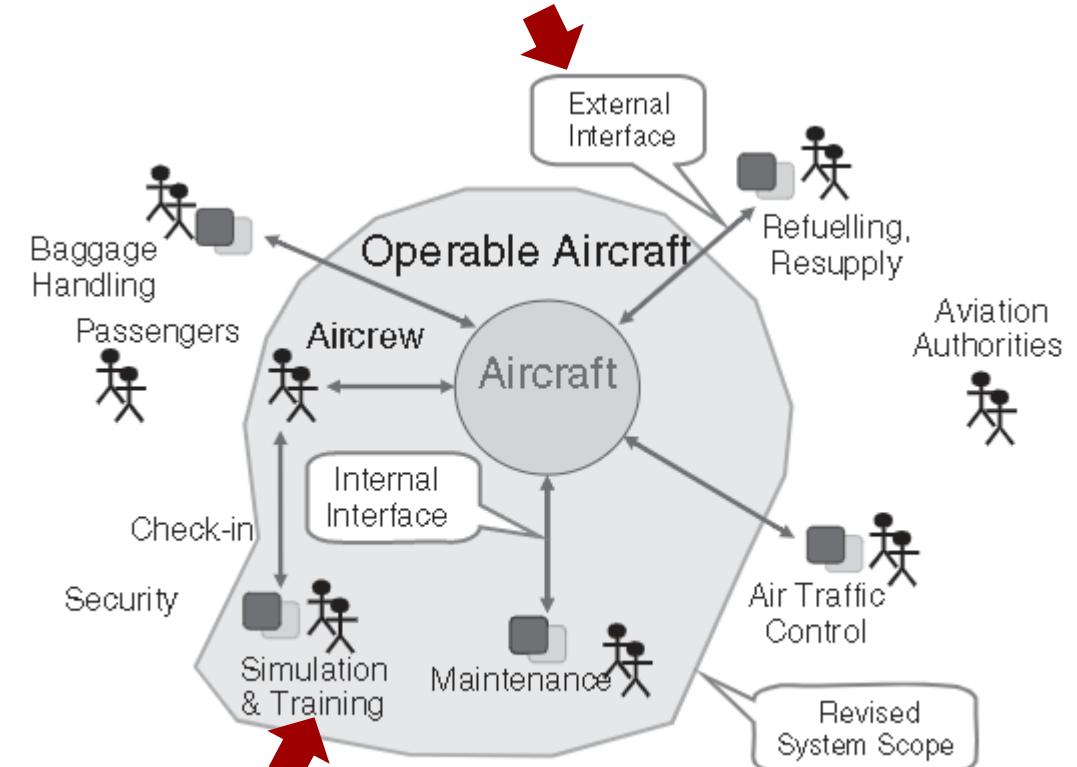
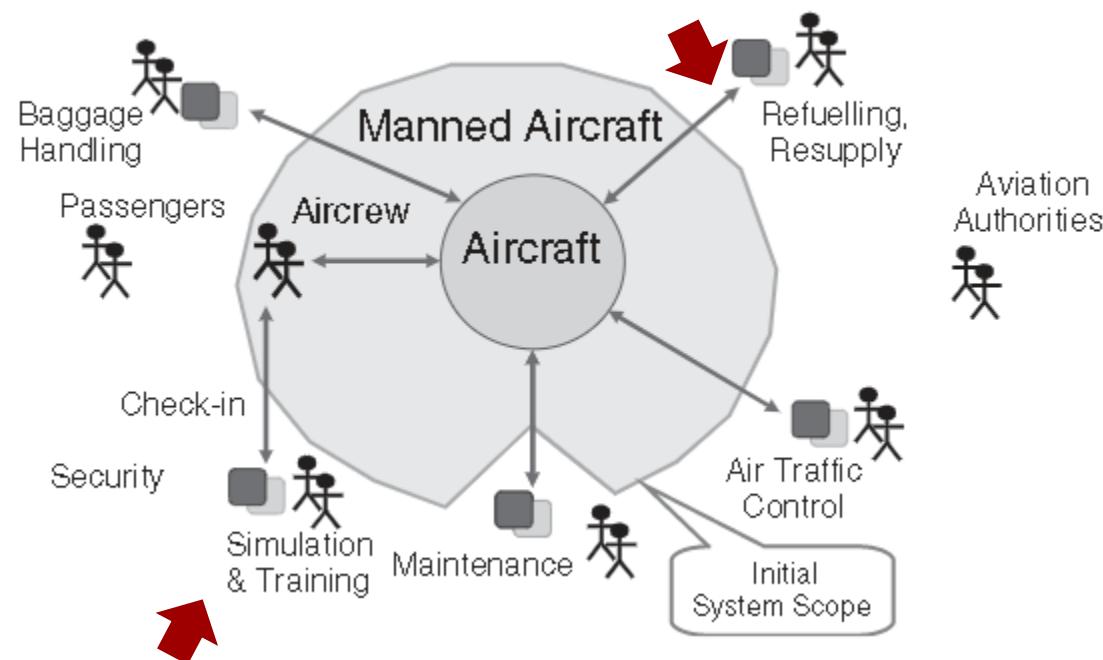
- Products
- People
- Procedures



A **system** is a construct or collection of different elements that together produce results not obtainable by the elements alone.

Example of Scope Definition

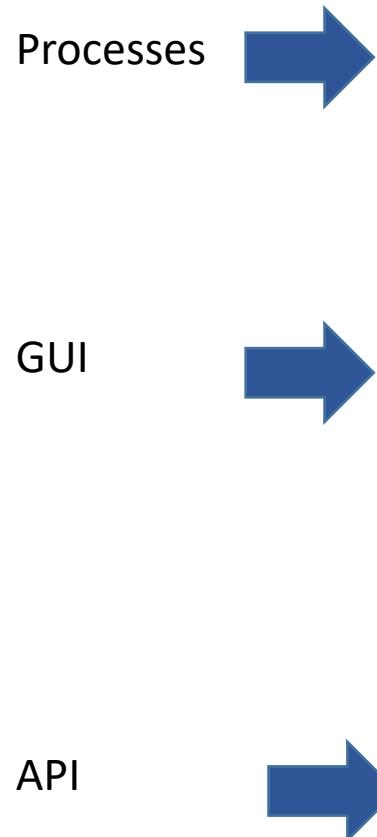
- Spot the differences in the two system representations
- Discuss on pros and cons of the two scopes



Identifying Interfaces

- Interface: a point where two systems, subjects, organizations, etc., meet and interact
- Interfaces are not necessarily hardware:
 - USB, Firewire
 - Gas tank and pump at gas stations
 - Mailbox
 - Point of contact
- Pay attention to interfaces among people (soft system)
- Create a context diagram

Interfaces and Requirements Elicitation Techniques

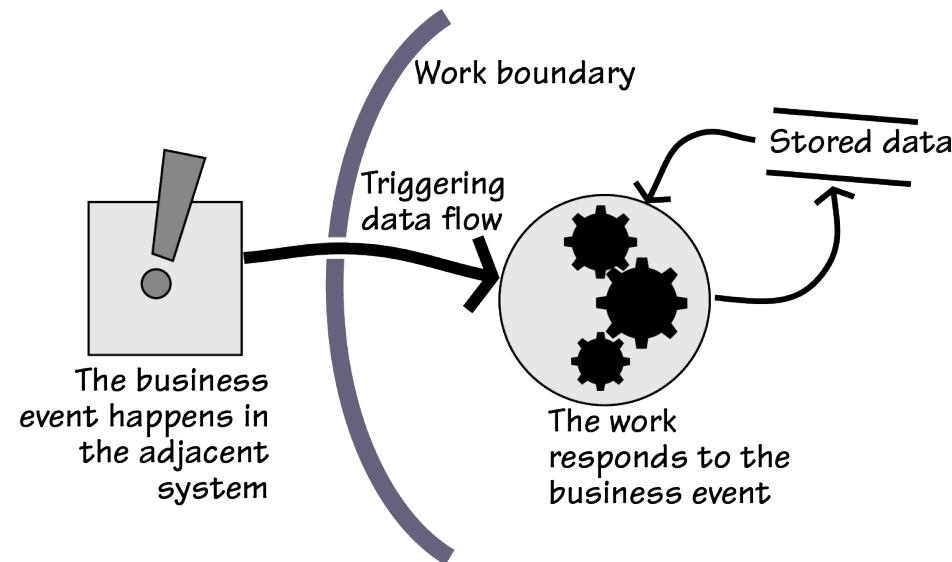


| Type of interface | Example | How to discover this | Treatment |
|--------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Between people | Aircrew report a problem to maintenance staff | Study the 'soft' system and its business processes (this chapter) Scenarios (Chapter 5) | Write a procedure; train crew to follow the procedure (i.e., write a training course, and run it) |
| Between people and products (These can be divided into inputs and outputs) | Aircrew interact with controls on the aircraft, both giving commands and receiving information | Scenarios (Chapter 5) Prototyping (Chapter 13) Similar/existing/rival products (Chapter 13) | Specify the user interface, considering effect on work (ergonomics); write a procedure for each use of the interface; train crew to follow the procedure (i.e., write a training course, and run it) |
| Between products (These can be in either direction, or both) | Automatic equipment diagnoses a fault on the aircraft | For interfaces to existing external systems: standards, or manufacturers' data sheets For interfaces to existing products/systems: speak to your developers/suppliers For new interfaces: iterate requirements and design (Chapter 14) | Specify the interface (the data and other quantities to be exchanged, and the required behaviour), using standards if possible; impose the interface as requirements on both products |

Scope and System Events

- Flows are provided as event that are enabled by the interfaces
- Draw or establish events on the boundary of the system
- An arrow into the circle on a context diagram means an event
 - Unpredictable arrival of a signal to which the **system has to respond by carrying out one or more tasks-> functional decomposition**
- It is a good practice to **start with a list of in/out events the system need to handle**
 - **(or the system needs to trigger, then handle...)**
- The scope consists on deciding the subset of events that will be handled by the system
- Handling such event become a requirement if agreed in the scope
- This has contractual implications for the product acceptance
- Events could be:
 - External: message, signal , control input
 - Time-triggered: time signal, polling cycle, etc.

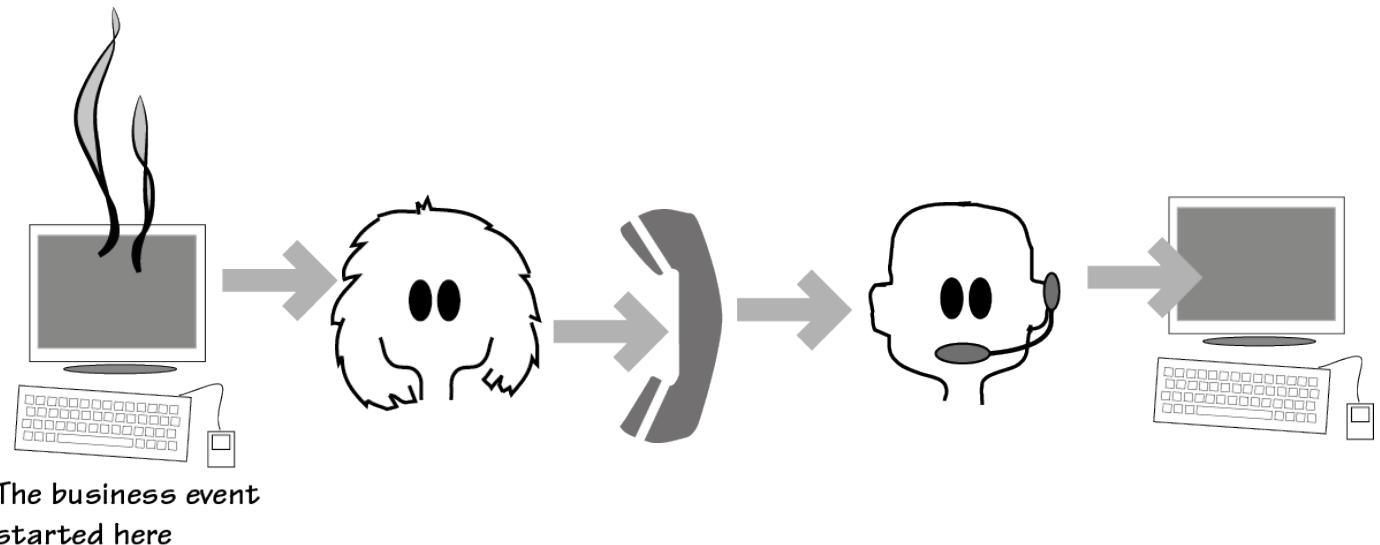
Flows are Triggered by Business Events (Ch4 Robertson)



Understanding of the sources of the business event allow to a better understanding of the system and to come up with innovative designs

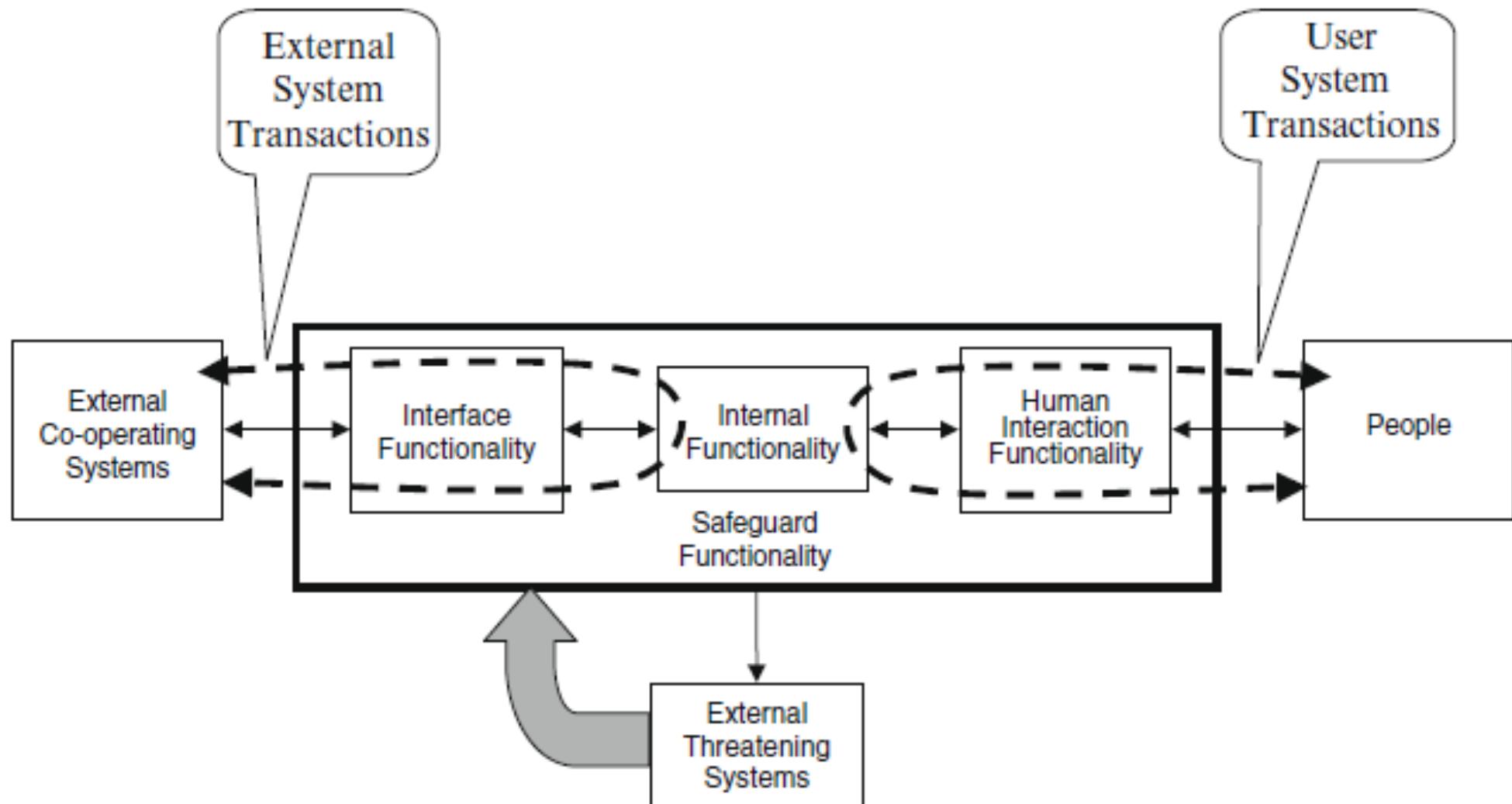
Business events can be time-based triggered (timers, or a monthly bill)

Establish the true actors of the system and flows by identifying business events



The maintenance call is received by the operator and input in the system *

Grouping Events Based on Interfaces (Hull, 2011)



Context Diagram and Events



| Event/Task | In/Out |
|---------------------------|--------------|
| Load Bags | In Scope |
| Unload Bags | In Scope |
| Refuel | In Scope |
| Check-in Passengers | Out of Scope |
| Radio Air Traffic Control | In Scope |
| Provide Airport Security | Out of Scope |
| | |
| | |

Event Handling Functions

- Express event handling by:
 - Textual event handling requirements
 - Condition-action tables
- Template for a traditional event handling requirement:
 - When<event is detected> do <required action>
 - Every <time period> do <required action>

When pressure in Main_Input_Pipe falls below Minimum_Input_Pressure, run Main_Pump at Full_Power.

Every Sample_Period, send Cylinder_Temperature to Engine_Controller.

When Payment_Confirmation is received, set Order_Status to To_Be_Despatched.

If Account_Balance exceeds Private_Banking_Threshold for Qualifying_Period then issue Private_Banking_Invitation to Customer.

Event Handling Functions(2)

■ Condition-Action Tables

- Condition action tables allow a more formal definition
- Conditions are expressed in columns (AND)
- Implication: conditions are mutually exclusive

| When condition₁ | and condition₂ | then |
|-----------------------------------------------------------------|----------------------------------|------------------------------------|
| <i>Pressure (Main_Input_Pipe) < Minimum_Input_Pressure</i> | | <i>Run Main_Pump at Full_Power</i> |
| <i>Pressure (Main_Input_Pipe) > = Minimum_Input_Pressure</i> | | <i>Main_Pump Inactive</i> |

Users and Scenarios

- Using the events and interfaces identified, lets generate user cases and system scenarios

As a <user role> when <event>, I want to be able to <capability> so that I can <goal>

| Requirement Elements | | Priorities |
|-------------------------|--|---------------------------|
| | | Measurements |
| | | Definitions |
| Discovery Contexts | | Rationale and Assumptions |
| Introduction | | Qualities and Constraints |
| From Individuals | | Scenarios |
| From Groups | | |
| From Things | | |
| Trade-Offs | | |
| Putting it all Together | | |

- Interfaces=>enable events
- Scope=collection of events
- System capabilities allow stakeholder to achieve goals



How do we achieve capabilities?

Grouping Actions into Scenarios

- The system has to take several actions to achieve a capability
- This lead to the idea of scenario
- A scenario is a sequence of actions

System Output

- The system under design generates events that affect the world outside its boundaries
- Examples include:
 - Sending a message , sounding an alarm, etc.
- For a function or capability to be provided to a human operator:
 - The <operator> shall be enabled to< do the required action>
- For a product function:
 - The <product> shall <do the required actions> <with abc performance>

From Context to Use Case Diagrams

- Context diagram are not exactly use cases
- Context diagrams show interfaces not functions
- Use cases lists all the high-level functions that will allow the actors to achieve their goals (through the use of the system of course)

Sometimes the system is in the interface

dexcom
CONTINUOUS GLUCOSE MONITORING



TANDEM[®]
DIABETES CARE

t:slim X2 Insulin Pump
Easy to use. Easy to love.



When integrated with Dexcom G6 CGM[‡]

LARGE COLOR TOUCHSCREEN
Easy to read, simple to use interface

WATERTIGHT CONSTRUCTION
Tested to 3 feet for 30 minutes (IPX7)

RECHARGEABLE BATTERY
Convenient charging via micro-USB port

CUSTOM SETTINGS
Create up to six Personal Profiles

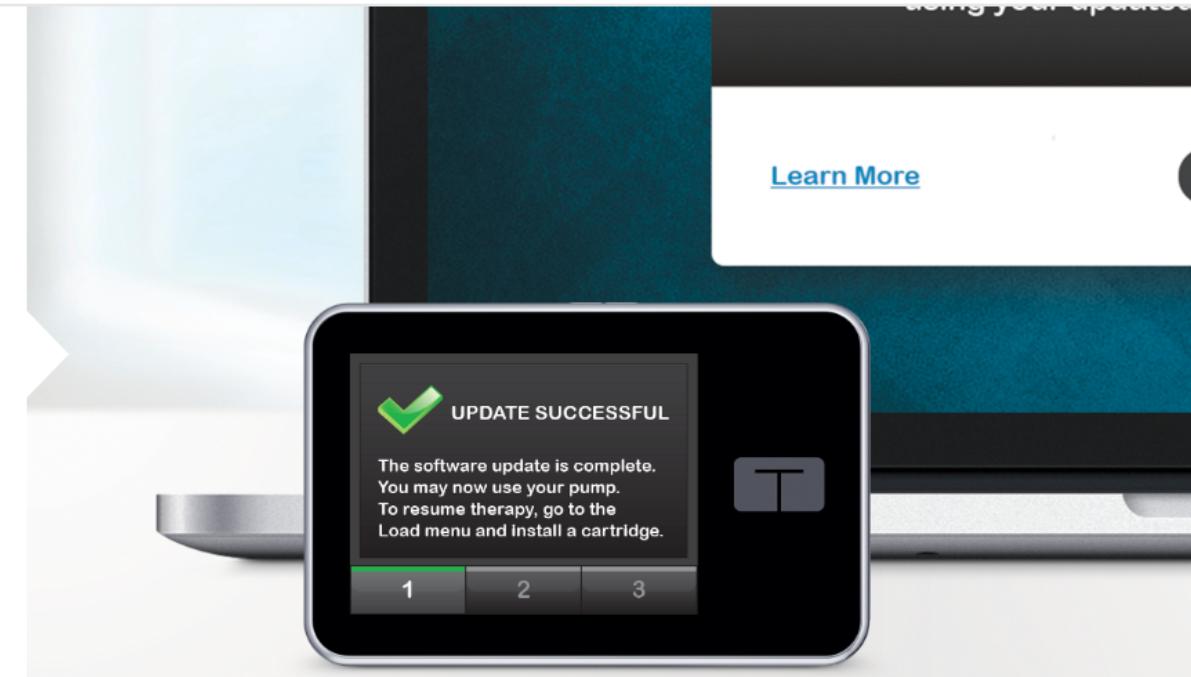
BLUETOOTH[®] WIRELESS TECHNOLOGY
Modern connectivity for a modern world

- Requirements exist for the lifecycle of the product

[PRODUCTS](#)[SUPPORT](#)[ABOUT](#)[CONTACT](#)[GET STARTED](#)

Remote software updates

The t:slim X2 insulin pump is the first FDA-approved insulin pump capable of remote feature updates.[†] Using a personal computer, patients can keep their pump up to date with the latest technology during its warranty period.



You know interfaces are important when...

Amazon, Apple, and Google unveil smart home collaboration



By [Pablo Valerio](#)

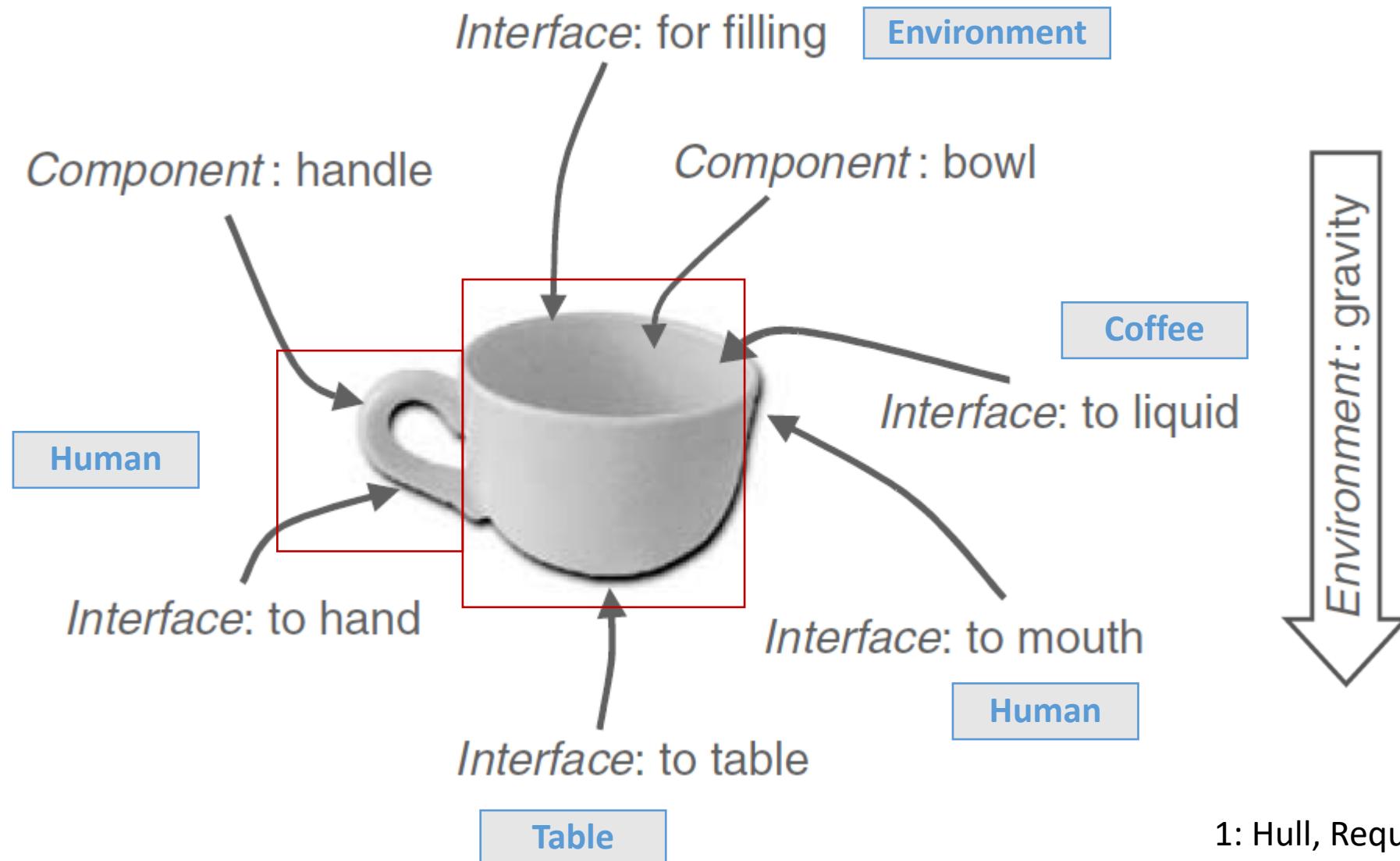
Posted on December 31, 2019



Interfaces and N-Squared Diagram

- The N-squared (N^2) diagram is used to determine system interfaces between components or functions.
- Rules:
 - The system components or functions are placed on the diagonal
 - Off-diagonal elements represent the interface inputs and outputs
 - Outputs are represented by rows
 - Inputs are represented by columns
 - Blanks mean no interface between the respective i-j components or functions
- Other usage
 - pinpoints areas where conflicts could arise in interfaces, and
 - highlights input and output dependency assumptions
- Not a formal SysML diagram

A Cup Viewed as a System¹



1: Hull, Requirements Engineering

- **Interface Requirements Document (IRD):** Defines the functional, performance, electrical, environmental, human, and physical requirements and constraints that exist at a common boundary between two or more functions or system elements
- **Interface Control Document or Interface Control Drawing (ICD):** Details the physical interface between two system elements, including the number and types of connectors, electrical parameters, mechanical properties, and environmental constraints.
- **Interface Definition Document (IDD) :** A unilateral document controlled by the end item provider. Provides the details of the interface for a design solution that is already established.

Validate Interfaces and Events

- For standard interfaces, ensure that the exact version of the relevant standard is identified
- For custom interfaces, agree all details of each interface with the owner of the other system
- Each event should be handled by when-requirements or their equivalent condition-action tables
- List of alternative conditions in condition-action tables cover all the possibilities exhaustively
- Check that each scenario referenced in the event handling requirement is defined as a use case
- Check that each interface is defined in the data dictionary

Group Activity

Get in new groups of 4



WHEREWELEARN

Demo Button Demo Button

Demo Button Demo Button Demo Button

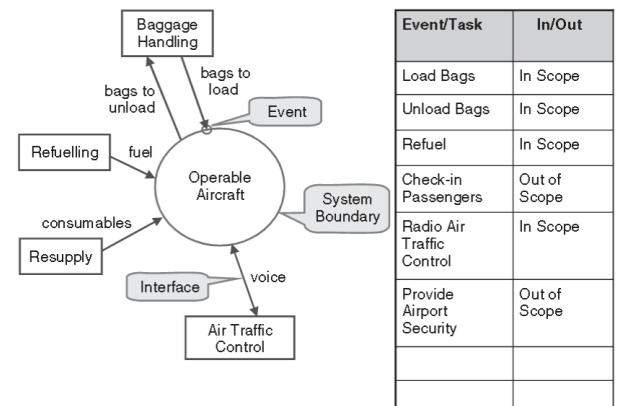
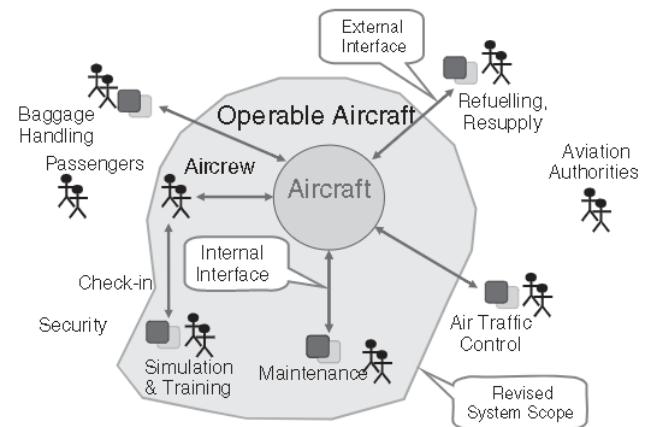
Demo Button Demo Button

MASTER SYSTEMS DISPLAY



In Class Exercise

- Create a Context Diagram and Events for the selected system.
- Include as many interfaces and events as you can identify
- Present to the class in under 3 min
- May use laptop or paper to present.



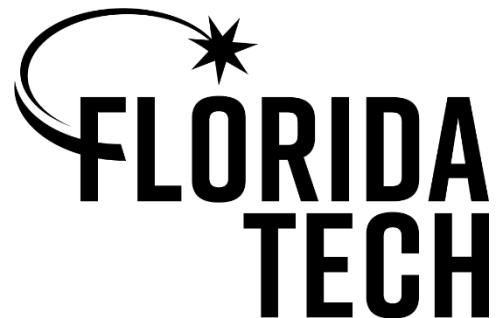
SYS 5460: Context , Interfaces, Scope

Contents

- Concept of Operations
- System Context
- Interfaces
- Examples

**Department of Computer &
Engineering Sciences**

College of Engineering
Florida Institute of Technology



PAPER REVIEW (20%)

- In teams of 2 students :
- Selected a published journal paper on a topic related to or discussed in class.
- Must be published in a JOURNAL
- Start Looking for Topic as soon as possible
- Submit topic (paper title and abstract) for approval
- Topics due 2 weeks from today
- Read and understand the paper, prepare a 5 min presentation on the topic (slides submitted via canvas 2 weeks after project acceptance)
- Present paper topic in class



How the customer
explained it



How the project leader
understood it



How the analyst
designed it



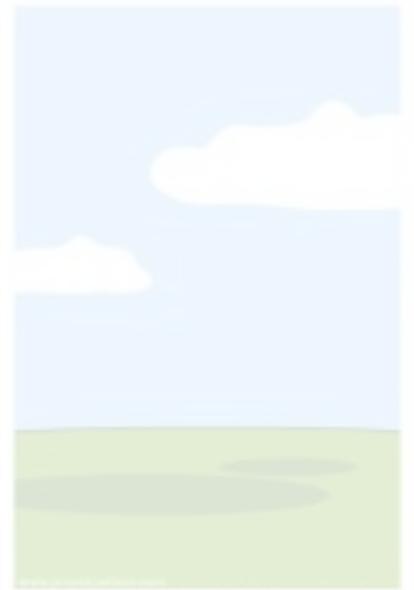
How the programmer
wrote it



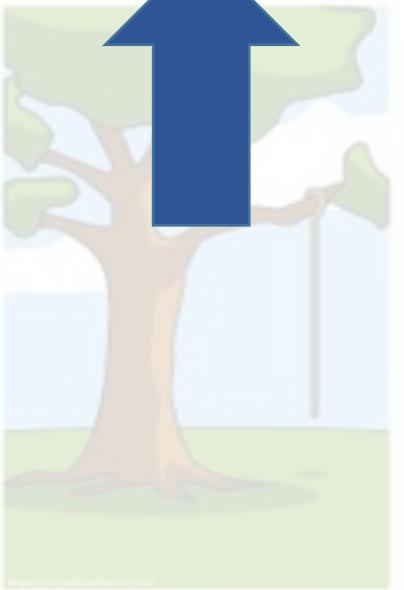
What the beta testers
received



How the business
consultant described it



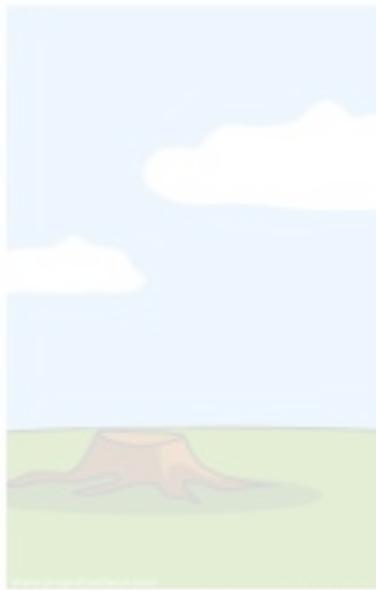
How the project was
documented



What operations
installed



How the customer was
billed



How it was supported



iSwing

What marketing
advertised

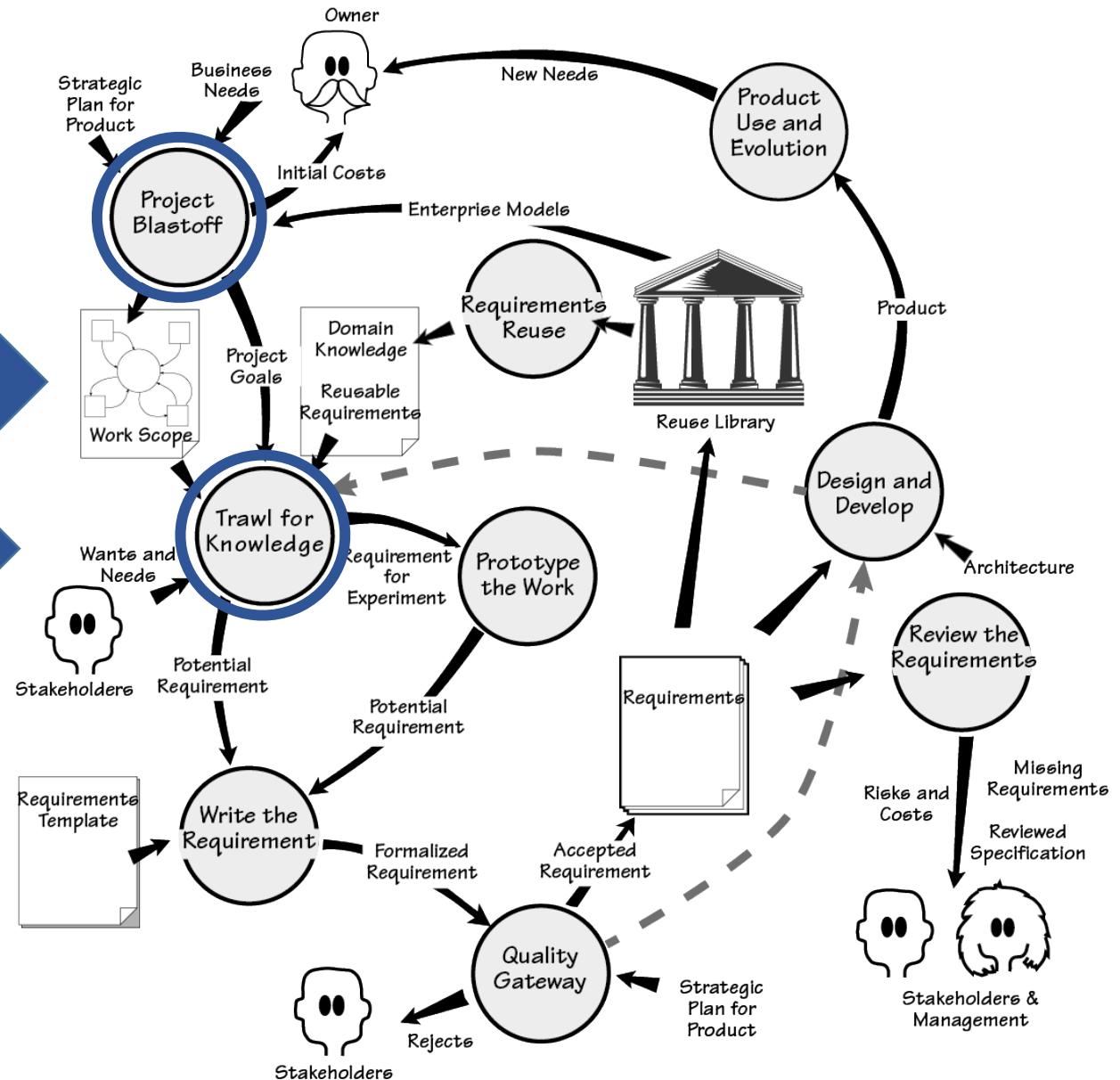


What the customer
really needed

Volere Requirements Process (Robertson)

Model system functionality

Concept of Operations



Standards For Concept of Operations

Previous standards

ANSI/AIAA-G-043 Outline

1. Scope
2. Referenced Documents
3. User-Oriented Operational Description
4. Operational Needs
5. System Overview
6. Operational Environment
7. Support Environment
8. Operational Scenarios

IEEE 1362 Outline

1. Scope
2. Referenced Documents
3. The Current System or Situation
4. Justification for and Nature of Changes
5. Concepts for the Proposed System
6. Operational Scenarios
7. Summary of Impacts
8. Analysis of the Proposed System



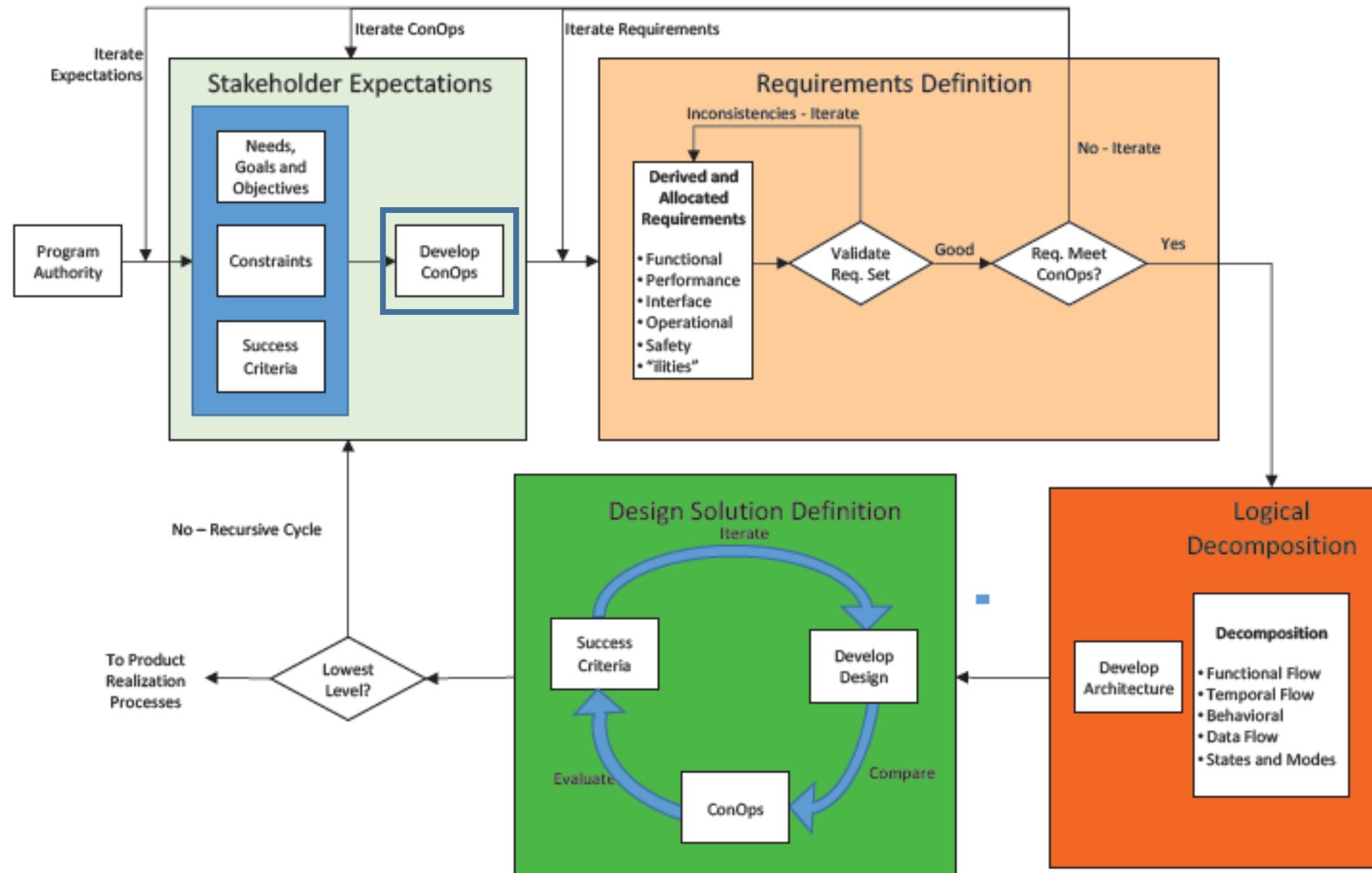
Current standards ISO/IEC 12207 Software lifecycle process or ISO/IEC 15288 Systems and Software lifecycle processes

concept of operations

verbal and/or graphic statement, in broad outline, of an organization's assumptions or intent in regard to an operation or series of operations

Note 1 to entry: The concept of operations frequently is embodied in long-range strategic plans and annual operational plans. In the latter case, the concept of operations in the plan covers a series of connected operations to be carried out simultaneously or in succession. The concept is designed to give an overall picture of the organization operations. See also operational concept.

Note 2 to entry: It provides the basis for bounding the operating space, system capabilities, interfaces and operating environment.



DoDAF Viewpoints and Models

OV-1: High-Level Operational Concept Graphic

OV-2: Operational Resource Flow Description

OV-3: Operational Resource Flow Matrix

OV-4: Organizational Relationships Chart

OV-5a: Operational Activity Decomposition Tree

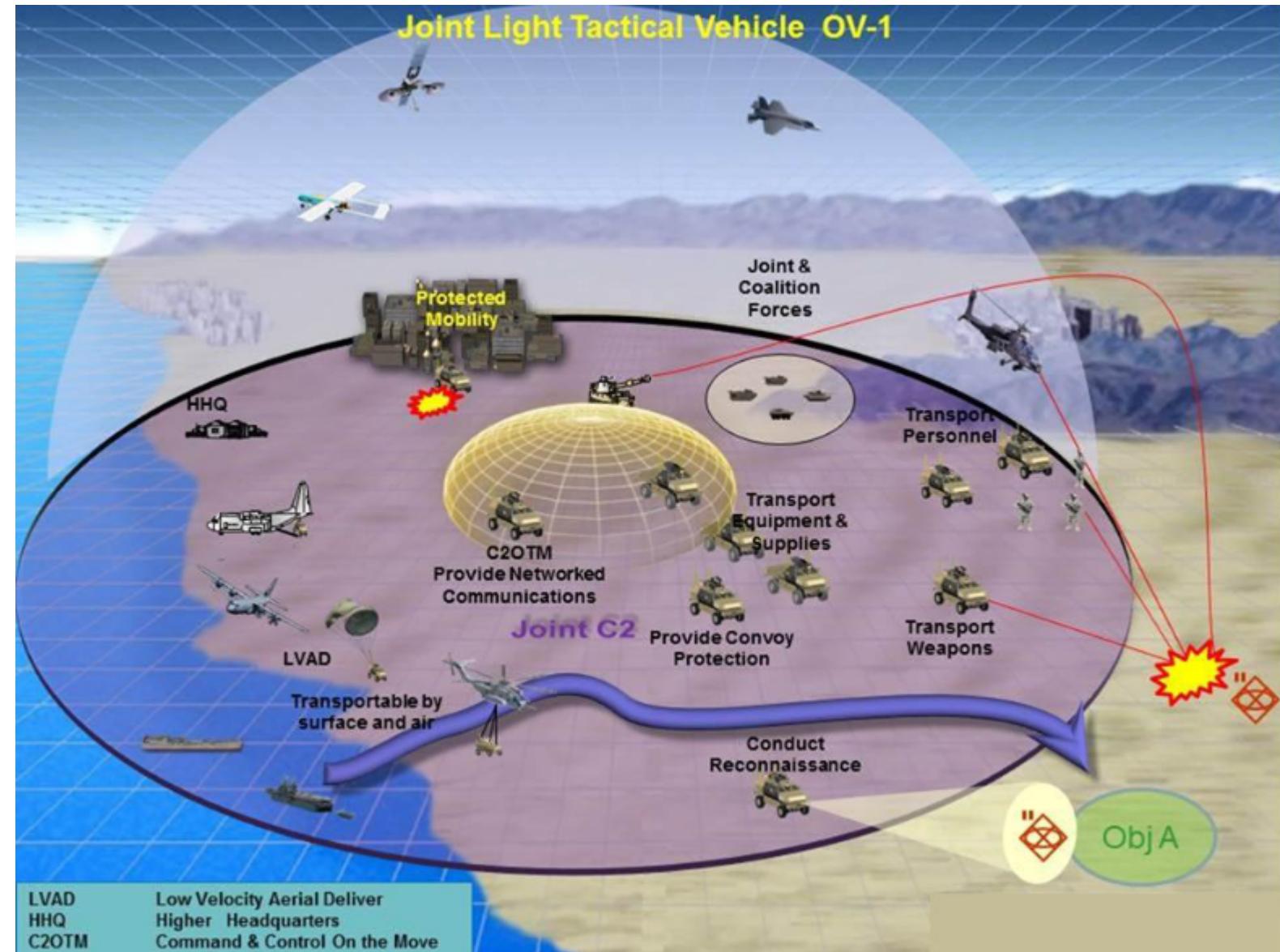
OV-5b: Operational Activity Model

OV-6a, 6b, 6c: Introduction

OV-6a: Operational Rules Model

OV-6b: State Transition Description

OV-6c: Event-Trace Description



Concept of Operations Document (IEEE 1362-1998)

Title page

Revision chart

Preface

Table of contents

List of figures

List of tables

1. Scope

 1.1 Identification

 1.2 Document overview

 1.3 System overview

2. Referenced documents

3. Current system or situation

 3.1 Background, objectives, and scope

 3.2 Operational policies and constraints

 3.3 Description of the current system or situation

 3.4 Modes of operation for the current system or situation

 3.5 User classes and other involved personnel

 3.6 Support environment

4. Justification for and nature of changes

 4.1 Justification of changes

 4.2 Description of desired changes

 4.3 Priorities among changes

 4.4 Changes considered but not included

5. Concepts for the proposed system

 5.1 Background, objectives, and scope

 5.2 Operational policies and constraints

 5.3 Description of the proposed system

 5.4 Modes of operation

 5.5 User classes and other involved personnel

 5.6 Support environment

6. Operational scenarios

7. Summary of impacts

 7.1 Operational impacts

 7.2 Organizational impacts

 7.3 Impacts during development

8. Analysis of the proposed system

 8.1 Summary of improvements

 8.2 Disadvantages and limitations

 8.3 Alternatives and trade-offs considered

9. Notes

Appendices

Glossary

ConOps

Project Drivers

1. The Purpose of the Project
2. The Stakeholders

Project Constraints

3. Mandated Constraints
4. Naming Conventions and Terminology
5. Relevant Facts and Assumptions

Functional Requirements

6. The Scope of the Work
7. The Business Data Model and Data Dictionary
8. The Scope of the Product
9. Functional Requirements

Non-functional Requirements

10. Look and Feel Requirements
11. Usability and Humanity Requirements
12. Performance Requirements
13. Operational and Environmental Requirements
14. Maintainability and Support Requirements
15. Security Requirements
16. Cultural Requirements
17. Legal Requirements

Project Issues

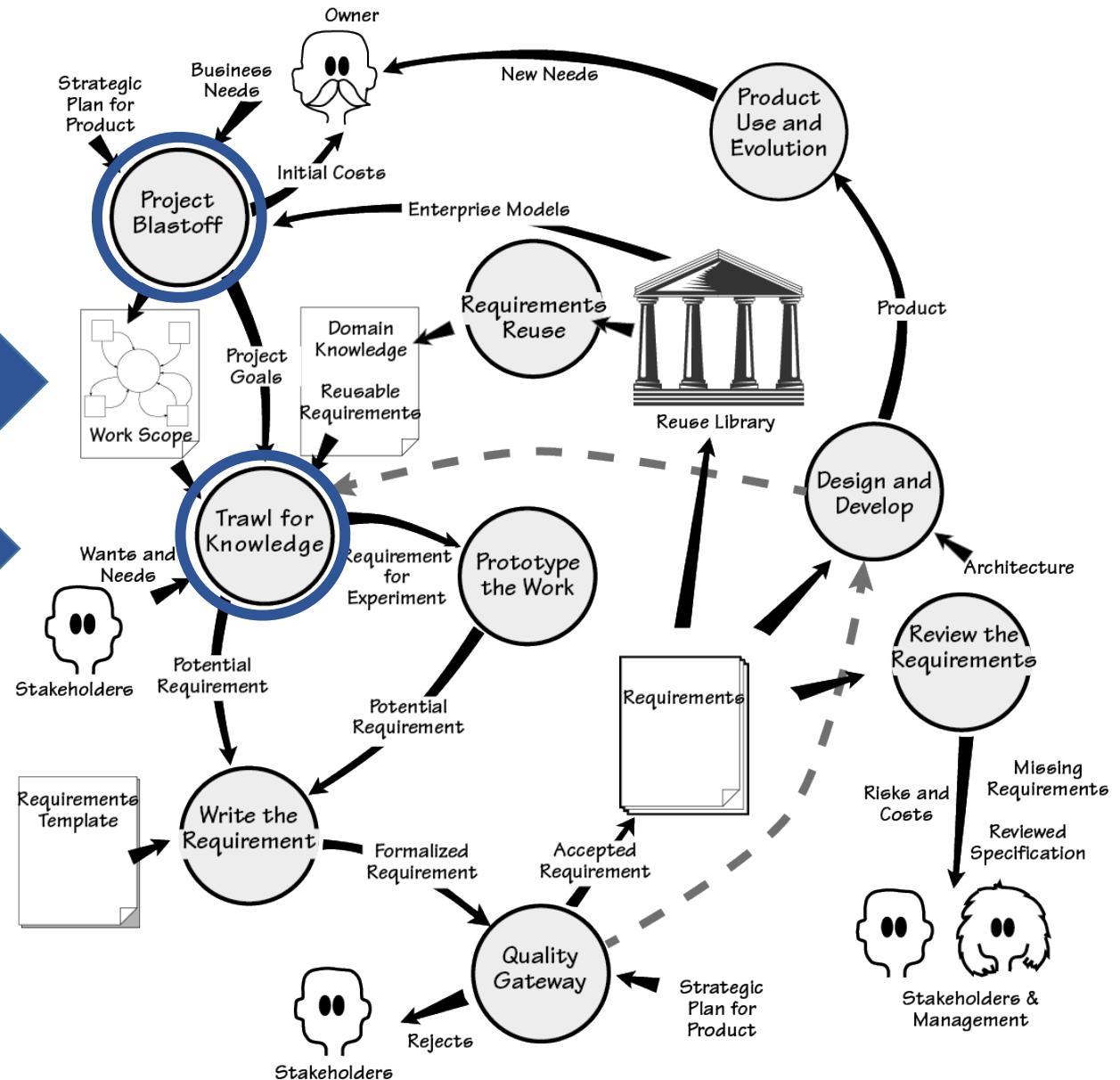
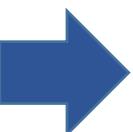
18. Open Issues
19. Off-the-Shelf Solutions
20. New Problems
21. Tasks
22. Migration to the New Product
23. Risks
24. Costs
25. User Documentation and Training
26. Waiting Room
27. Ideas for Solutions

Volere Requirements Process (Robertson)

Model system functionality



Concept of Operations



Project Blastoff (kickoff)

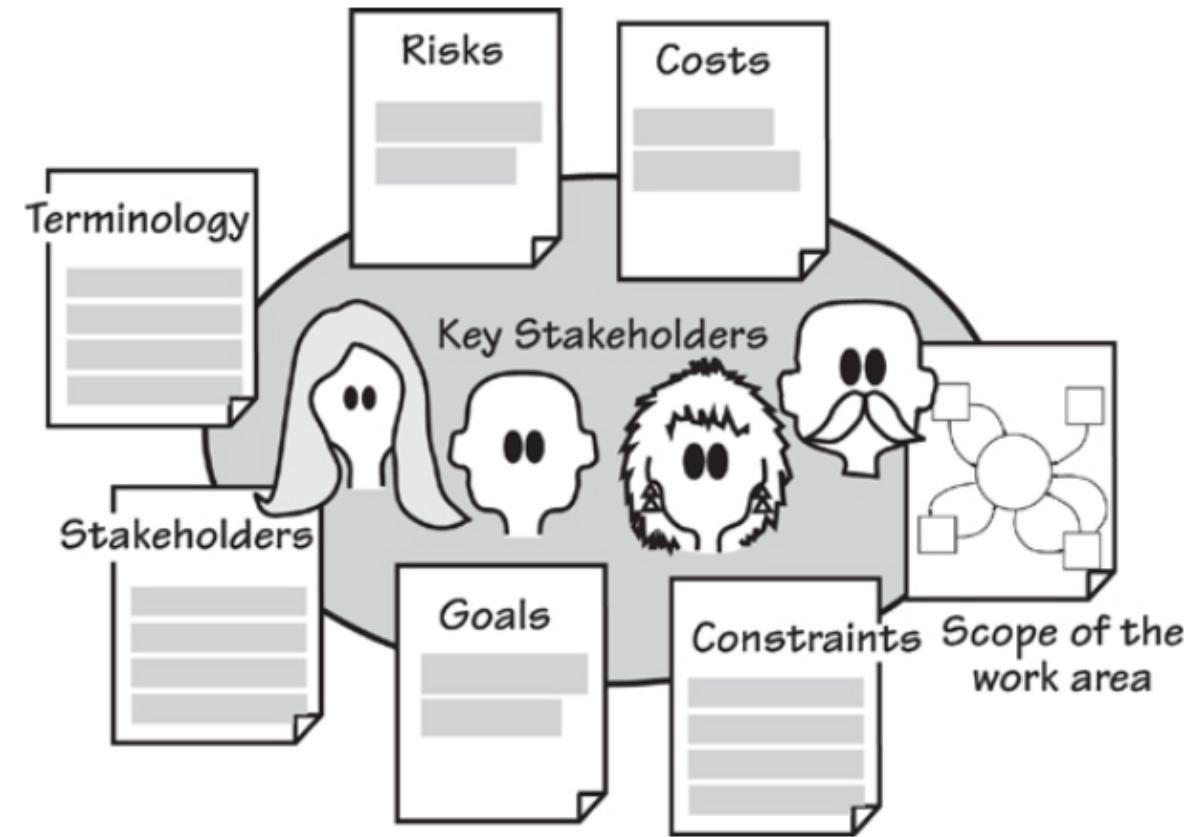
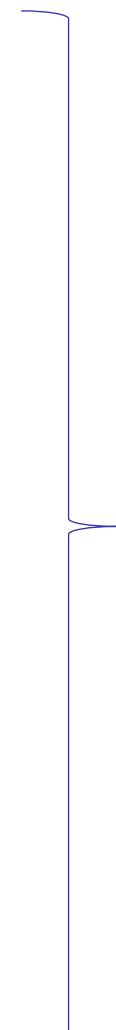
- The key purpose of project blastoff is to build the foundation for the requirements discovery, and to ensure that all the needed components for a **successful project** are in place.

- Involvement:

- Principal Stakeholder (sponsor)
- Key users
- Lead requirement analyst
- Technical business experts

- ▶ Discussions:

- Purpose of the project
- Scope of the work
- Stakeholders
- Constraints
- Included/excluded functionalities
- Special terminology
- Facts & Assumptions
- Cost estimates
- Risks



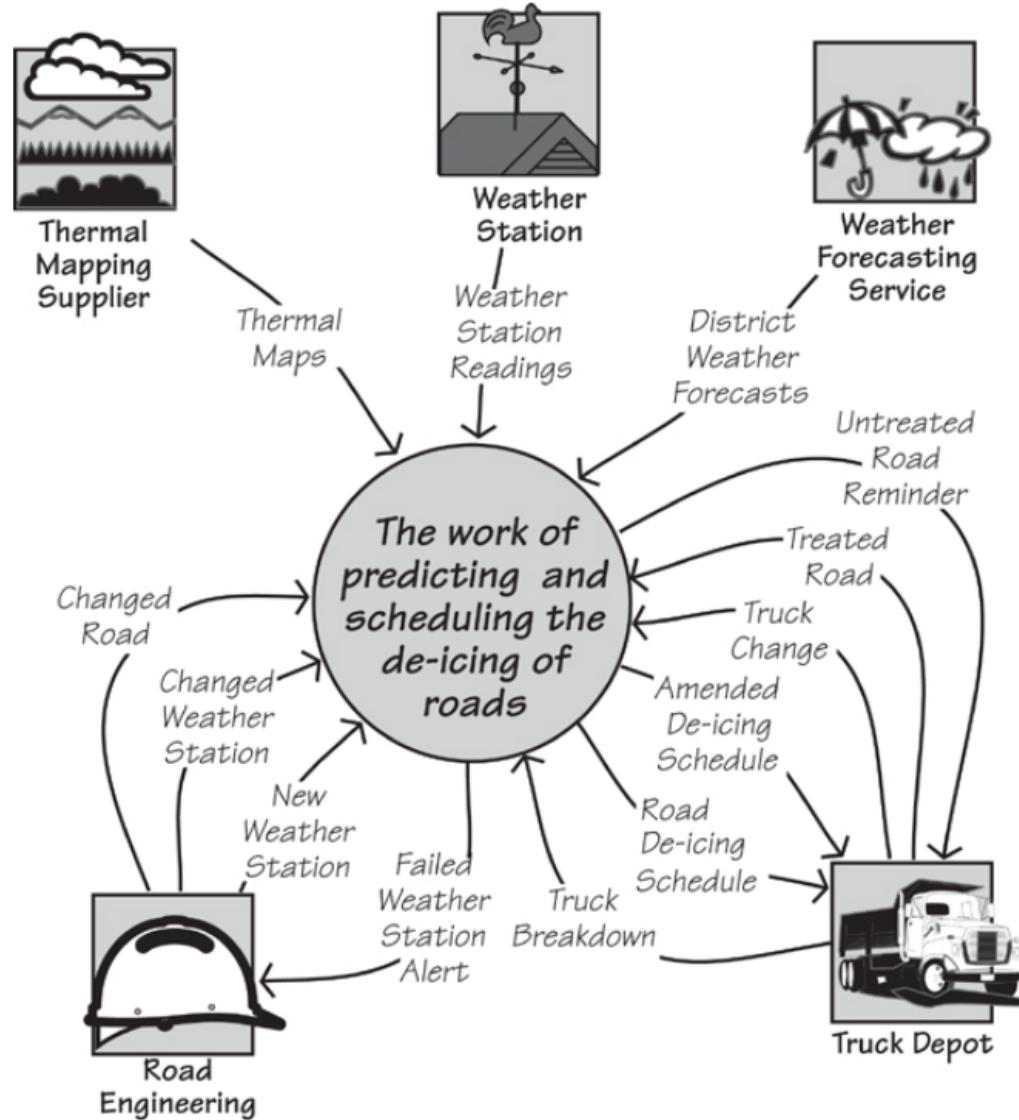
Importance of Product Scope

- A system react to external events
- Boundaries are not always well defined (soft systems, vs hard systems)
- Without boundaries scope creep is likely to occur
- Take time up front early in the project to define the scope of the product
- Understand the product context and interfaces with enough detail to be able to make accurate scoping decisions
- For some projects or program this may influence parts of the work and requirements that are “flown down” to external entities

Scoping the system allows separating the **Work** from its **Environment**

To achieve the optimal value for the owner, study enough of the owner's work to identify what is valuable (Robertson)

Context Diagram (Robertson)



The work context shows where the responsibilities of the work and the responsibilities of the adjacent systems start and end.

the responsibilities are defined by the flows of data on the context diagram. Always ask ...Why is this flow there?

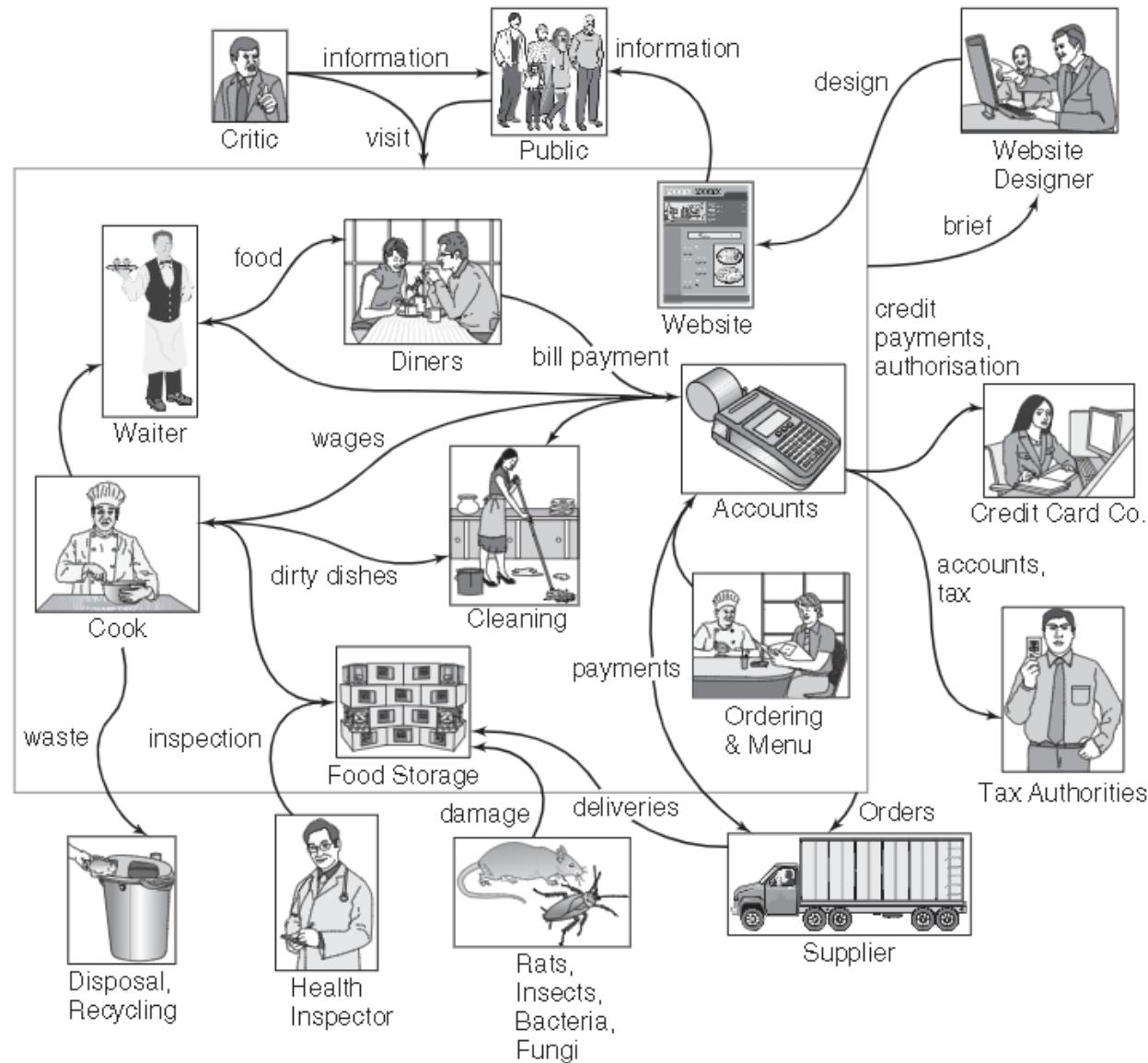
The context diagram intentionally ignores anything that happens outside the boundary that does not directly involve an interface with the system in question

Move from an soft system problem to a hard system problem

Soft System

- A soft system involves social, political and emotional issues as well as technology.
- Soft systems address messy , ill-structured problem situations
- A hard system may be in place , but it may fail for soft reasons:
 - IT solution to provide support, may fail because too complicated for some users
- Once implemented a product or service become part of the system
- Narrow system boundaries-> hard system
- Wide boundaries->soft system
- Draw a rich picture of the system showing:
 - Stakeholder not directly involved in the operations
 - Issues and concerns
 - Processes of stakeholders
 - Negative aspects
- The system boundary is outside the product boundary

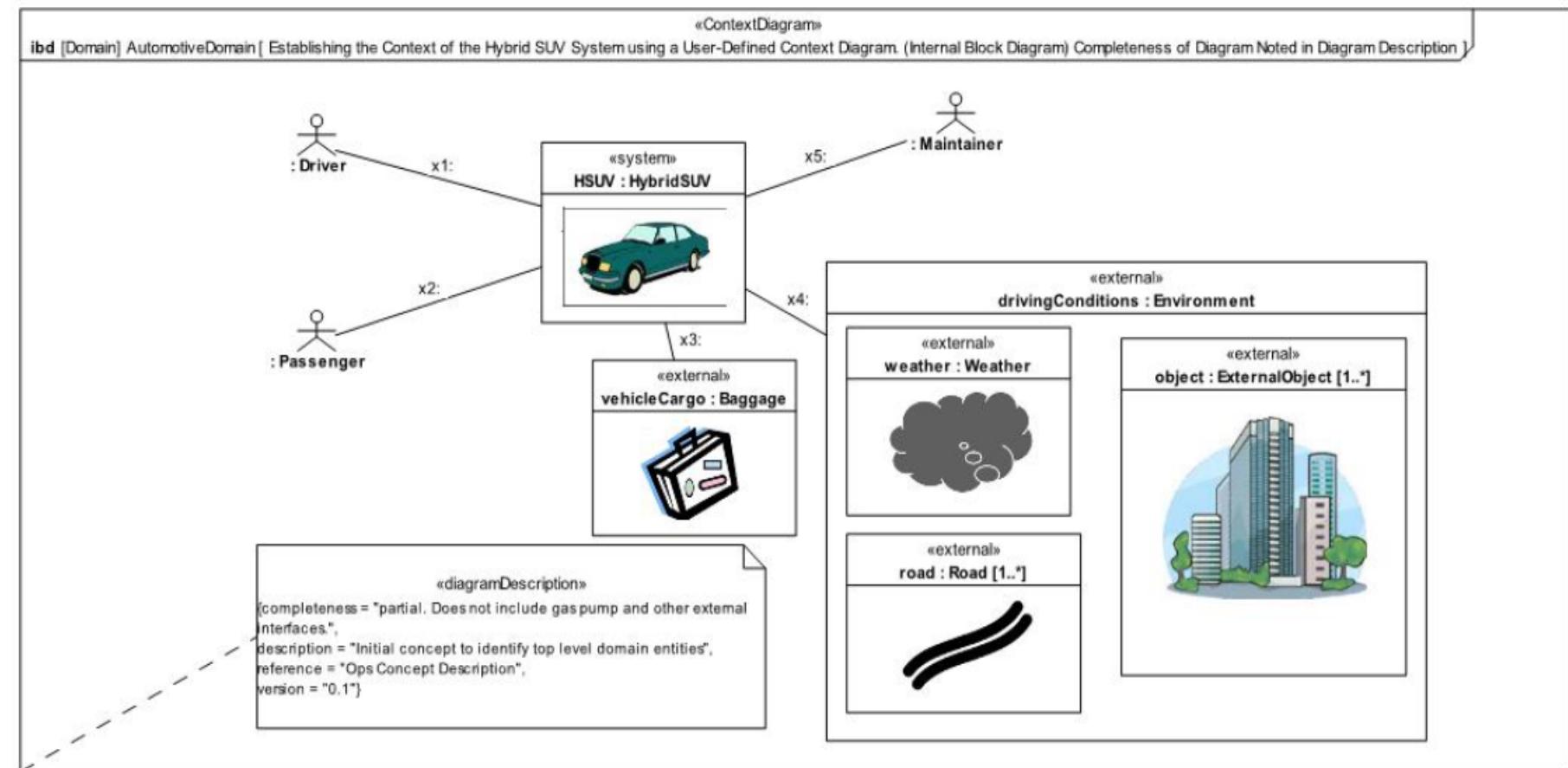
Example: Rich Picture of a Restaurant



What will be the system , if your product is the restaurant accounts?

Context Diagram of an SUV (SysML 1.6 Reference Document)

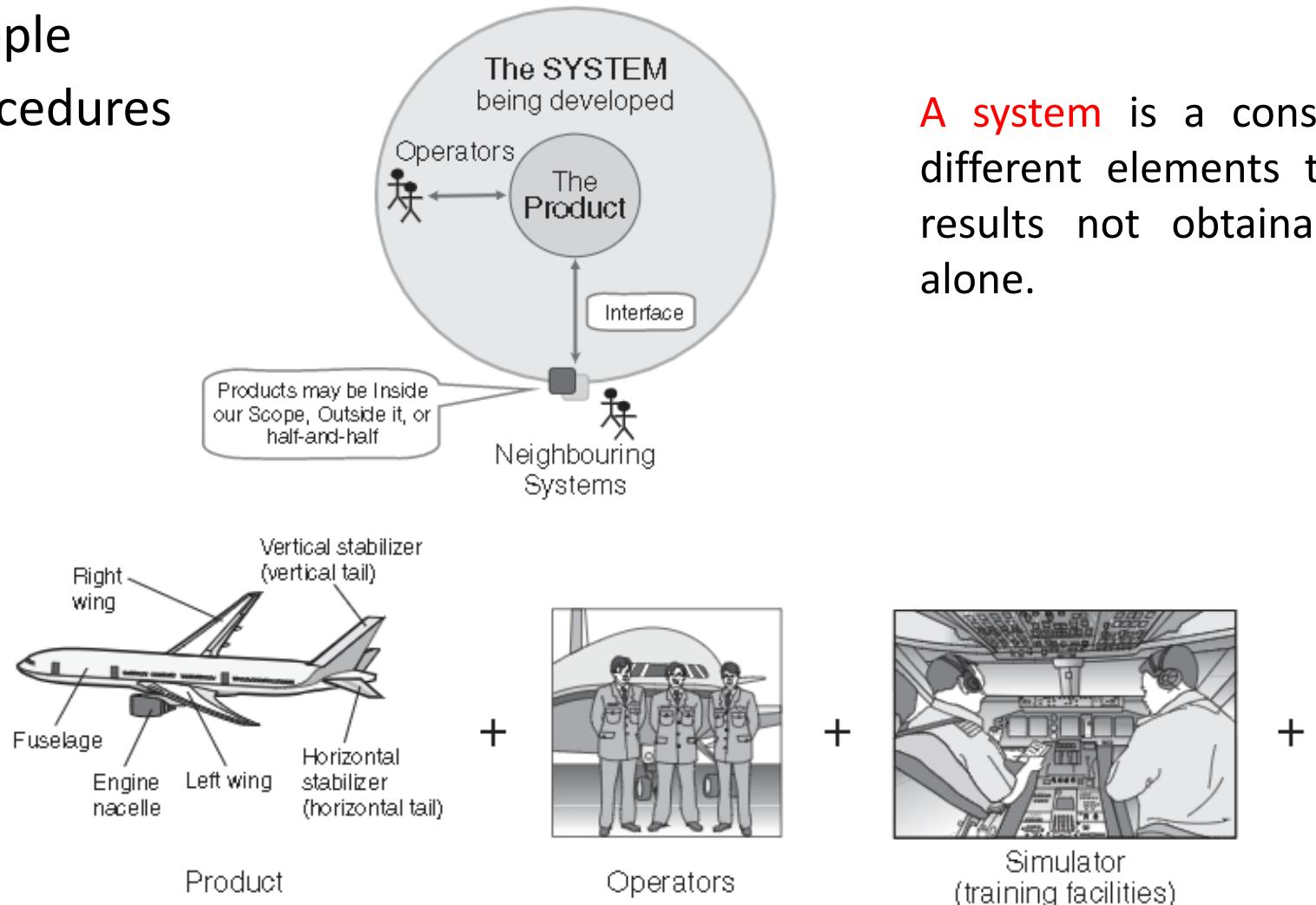
- An actor is an entity that interact with system
- Actors are outside the boundaries of the system
- Actors could be:
 - Hardware devices
 - Legacy systems
 - Other subsystems
 - Human users



System and its Context

- System consists on several components working together

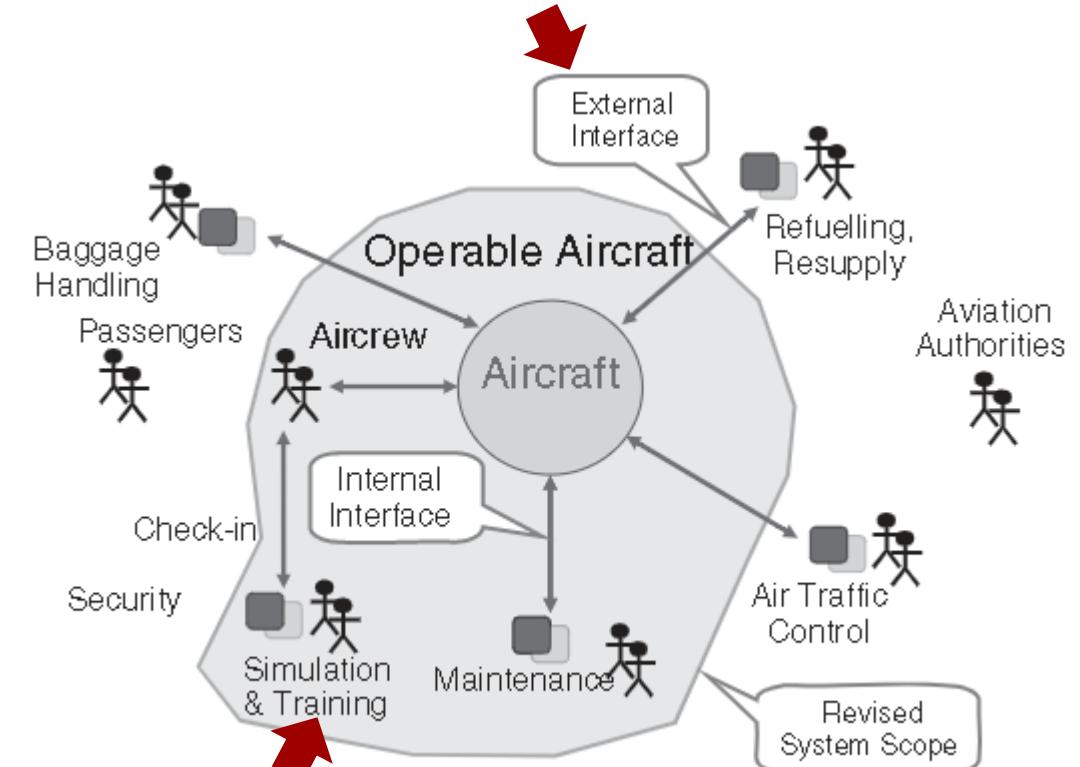
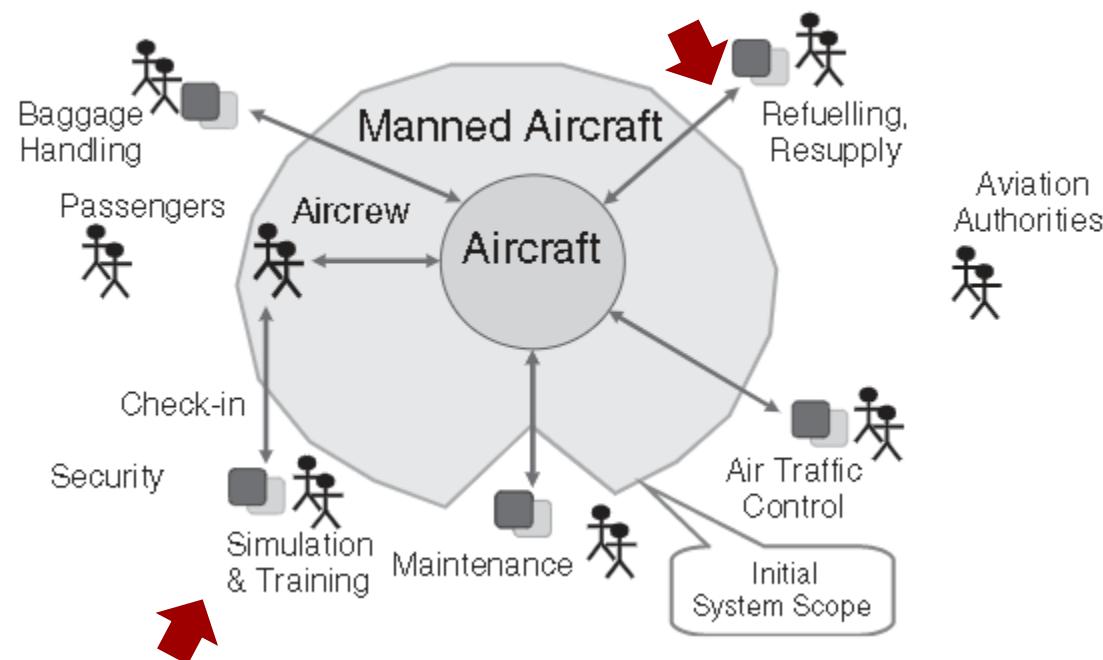
- Products
 - People
 - Procedures



A **system** is a construct or collection of different elements that together produce results not obtainable by the elements alone.

Example of Scope Definition

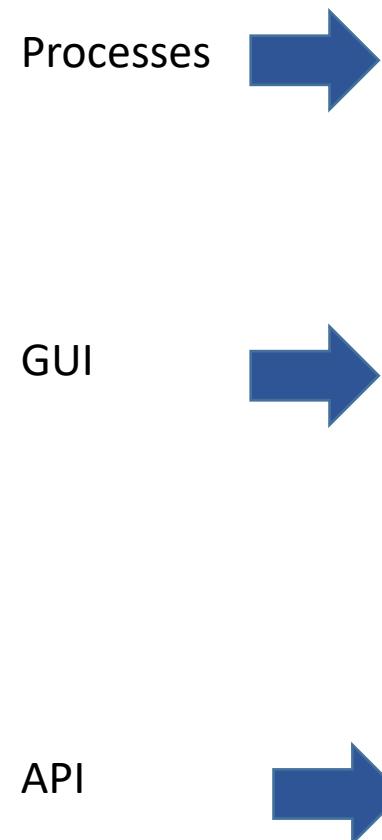
- Spot the differences in the two system representations
- Discuss on pros and cons of the two scopes



Identifying Interfaces

- Interface: a point where two systems, subjects, organizations, etc., meet and interact
- Interfaces are not necessarily hardware:
 - USB, Firewire
 - Gas tank and pump at gas stations
 - Mailbox
 - Point of contact
- Pay attention to interfaces among people (soft system)
- Create a context diagram

Interfaces and Requirements Elicitation Techniques

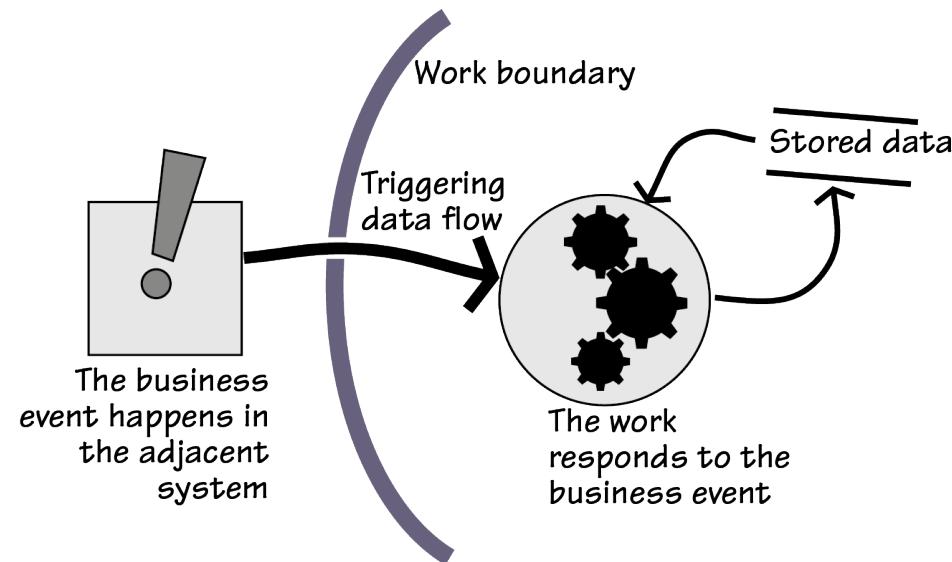


| Type of interface | Example | How to discover this | Treatment |
|--------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Between people | Aircrew report a problem to maintenance staff | Study the 'soft' system and its business processes (this chapter) Scenarios (Chapter 5) | Write a procedure; train crew to follow the procedure (i.e., write a training course, and run it) |
| Between people and products (These can be divided into inputs and outputs) | Aircrew interact with controls on the aircraft, both giving commands and receiving information | Scenarios (Chapter 5) Prototyping (Chapter 13) Similar/existing/rival products (Chapter 13) | Specify the user interface, considering effect on work (ergonomics); write a procedure for each use of the interface; train crew to follow the procedure (i.e., write a training course, and run it) |
| Between products (These can be in either direction, or both) | Automatic equipment diagnoses a fault on the aircraft | For interfaces to existing external systems: standards, or manufacturers' data sheets For interfaces to existing products/systems: speak to your developers/suppliers For new interfaces: iterate requirements and design (Chapter 14) | Specify the interface (the data and other quantities to be exchanged, and the required behaviour), using standards if possible; impose the interface as requirements on both products |

Scope and System Events

- Flows are provided as event that are enabled by the interfaces
- Draw or establish events on the boundary of the system
- An arrow into the circle on a context diagram means an event
 - Unpredictable arrival of a signal to which the **system has to respond by carrying out one or more tasks-> functional decomposition**
- It is a good practice to **start with a list of in/out events the system need to handle**
 - **(or the system needs to trigger, then handle...)**
- The scope consists on deciding the subset of events that will be handled by the system
- Handling such event become a requirement if agreed in the scope
- This has contractual implications for the product acceptance
- Events could be:
 - External: message, signal , control input
 - Time-triggered: time signal, polling cycle, etc.

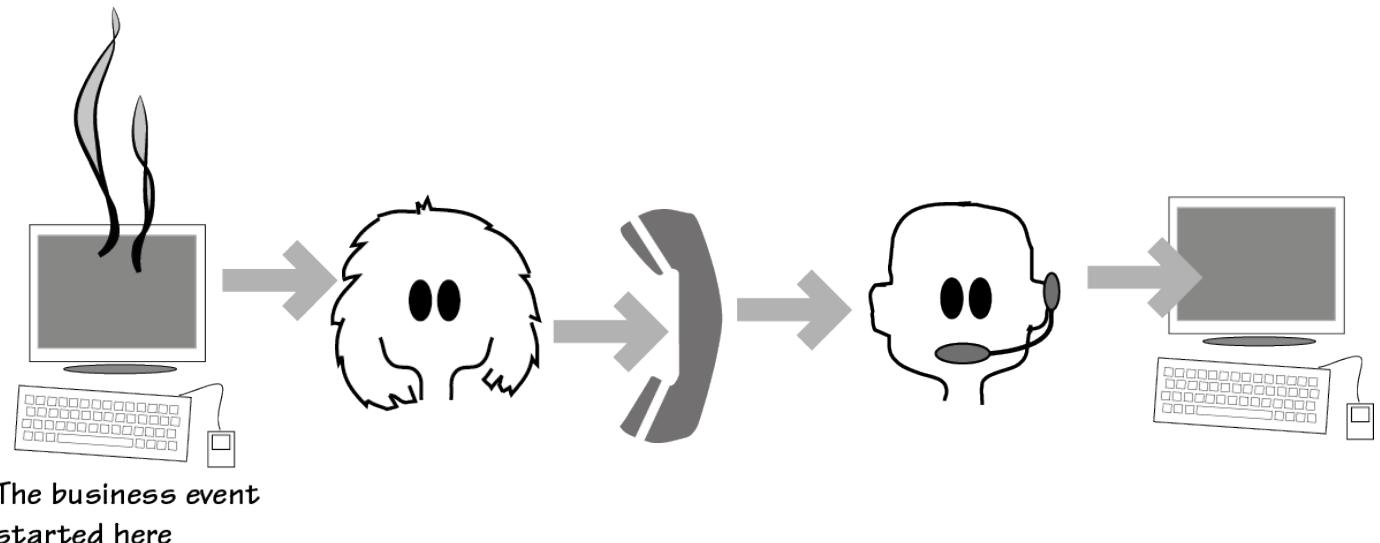
Flows are Triggered by Business Events (Ch4 Robertson)



Understanding of the sources of the business event allow to a better understanding of the system and to come up with innovative designs

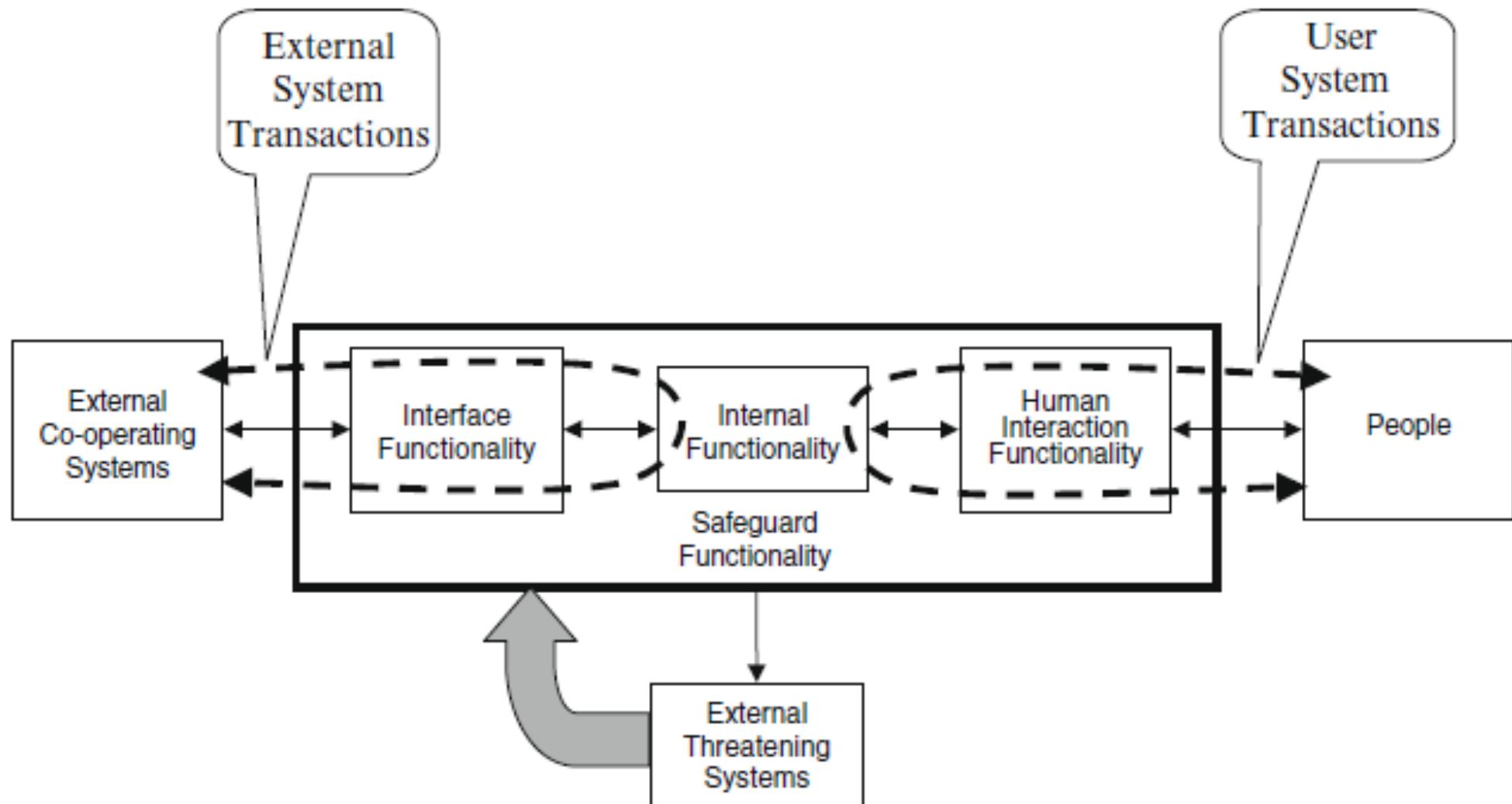
Business events can be time-based triggered (timers, or a monthly bill)

Establish the true actors of the system and flows by identifying business events



The maintenance call is received by the operator and input in the system *

Grouping Events Based on Interfaces (Hull, 2011)



Context Diagram and Events



| Event/Task | In/Out |
|---------------------------|--------------|
| Load Bags | In Scope |
| Unload Bags | In Scope |
| Refuel | In Scope |
| Check-in Passengers | Out of Scope |
| Radio Air Traffic Control | In Scope |
| Provide Airport Security | Out of Scope |
| | |
| | |

Event Handling Functions

- Express event handling by:
 - Textual event handling requirements
 - Condition-action tables
- Template for a traditional event handling requirement:
 - When<event is detected> do <required action>
 - Every <time period> do <required action>

When pressure in Main_Input_Pipe falls below Minimum_Input_Pressure, run Main_Pump at Full_Power.

Every Sample_Period, send Cylinder_Temperature to Engine_Controller.

When Payment_Confirmation is received, set Order_Status to To_Be_Despatched.

If Account_Balance exceeds Private_Banking_Threshold for Qualifying_Period then issue Private_Banking_Invitation to Customer.

Event Handling Functions(2)

- Condition-Action Tables

- Condition action tables allow a more formal definition
- Conditions are expressed in columns (AND)
- Implication: conditions are mutually exclusive

| When condition₁ | and condition₂ | then |
|-----------------------------------------------------------------|----------------------------------|------------------------------------|
| <i>Pressure (Main_Input_Pipe) < Minimum_Input_Pressure</i> | | <i>Run Main_Pump at Full_Power</i> |
| <i>Pressure (Main_Input_Pipe) > = Minimum_Input_Pressure</i> | | <i>Main_Pump Inactive</i> |

Users and Scenarios

- Using the events and interfaces identified, let's find out :
 - User Cases
 - Capabilities
 - Scenarios

As a <user role> when <event>, I want to be able to <capability> so that I can <goal>

| Requirement Elements | Priorities |
|-------------------------|----------------------------|
| Discovery Contexts | Measurements |
| Introduction | Definitions |
| From Individuals | Rationale and Assumptions |
| From Groups | Qualities and Constraints |
| From Things | Scenarios |
| Trade-Offs | Context, Interfaces, Scope |
| Putting it all Together | Goals |
| Stakeholders | Goals |

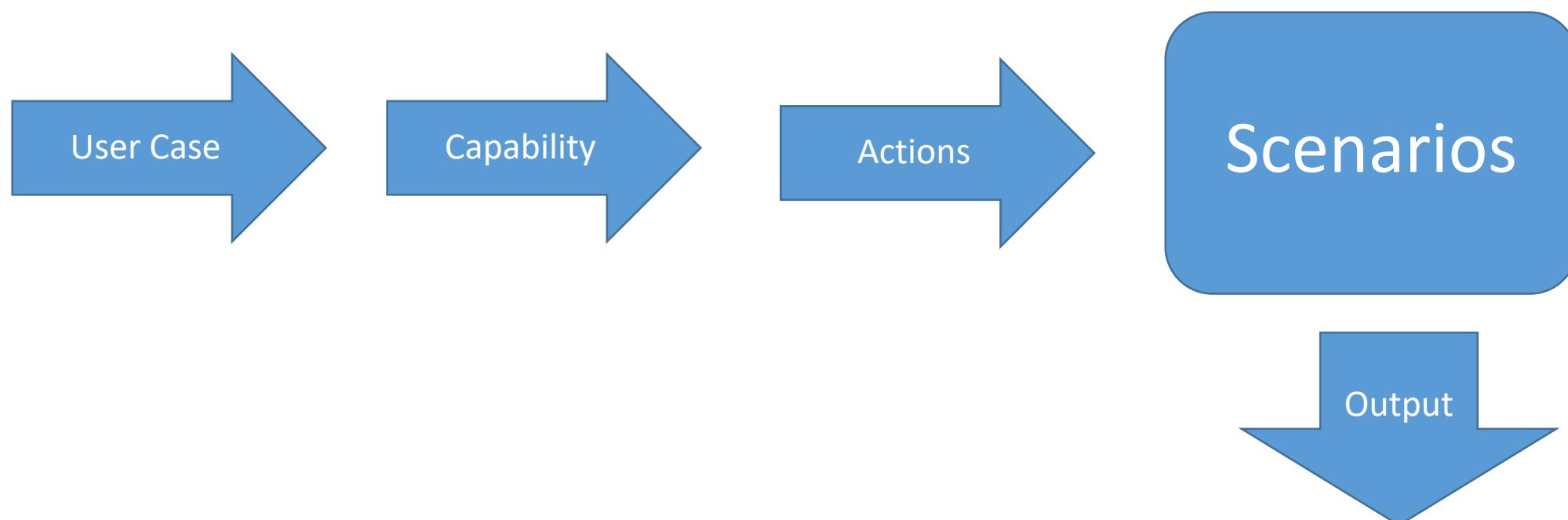
- Interfaces=>enable events
- Scope=collection of events
- System capabilities allow stakeholder to achieve goals



How do we achieve capabilities?

Grouping Actions into Scenarios

- The system has to take several actions to achieve a capability
- This lead to the idea of scenario
- A scenario is a sequence of actions



System Output

- The system under design generates events that affect the world outside its boundaries
- Examples include:
 - Sending a message , sounding an alarm, etc.
- For a function or capability to be provided to a human operator:
 - The <operator> shall be enabled to< do the required action>
- For a product function:
 - The <product> shall <do the required actions> <with abc performance>

From Context to Use Case Diagrams

- Context diagram are not exactly use cases
- Context diagrams show interfaces not functions
- Use cases lists all the high-level functions that will allow the actors to achieve their goals (through the use of the system of course)

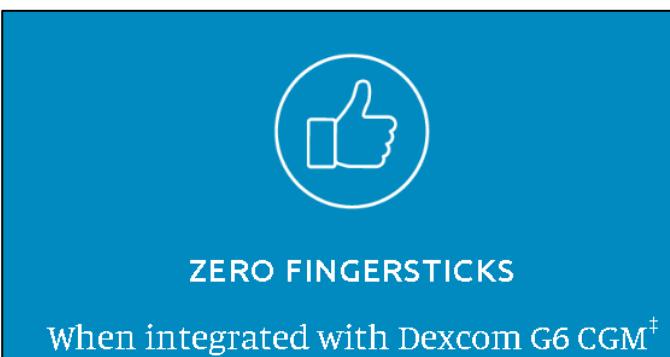
Sometimes the system is in the interface

dexcom
CONTINUOUS GLUCOSE MONITORING



TANDEM®
DIABETES CARE

t:slim X2 Insulin Pump
Easy to use. Easy to love.

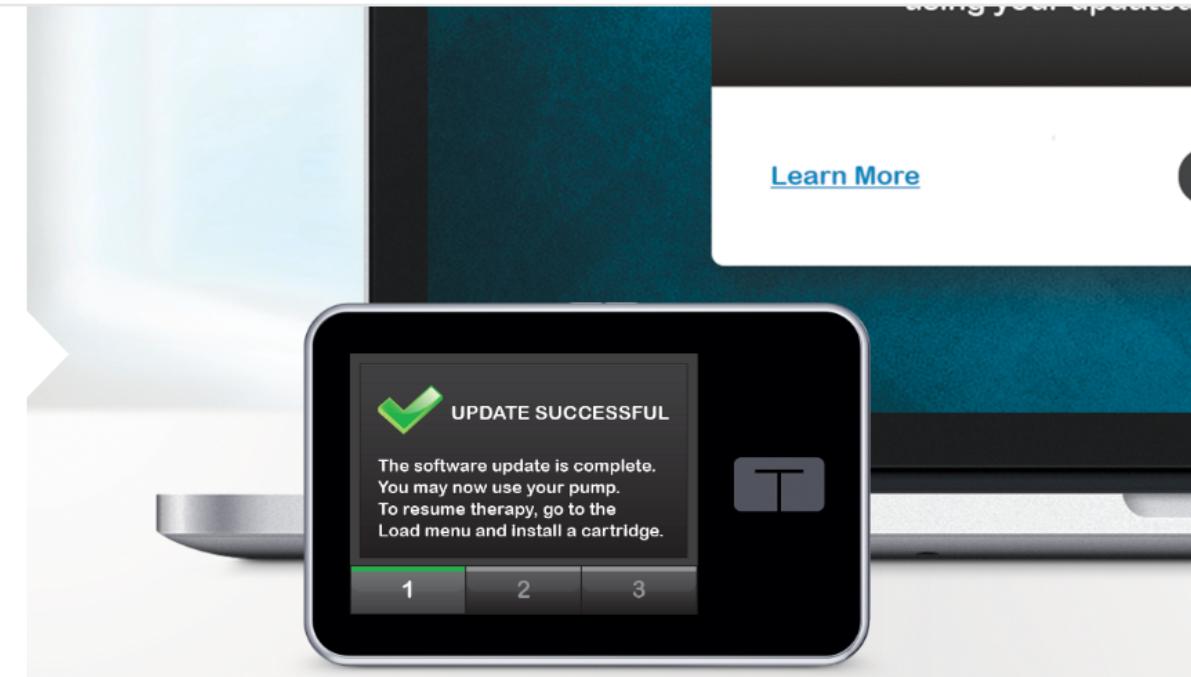


- Requirements exist for the lifecycle of the product

[PRODUCTS](#)[SUPPORT](#)[ABOUT](#)[CONTACT](#)[GET STARTED](#)

Remote software updates

The t:slim X2 insulin pump is the first FDA-approved insulin pump capable of remote feature updates.[†] Using a personal computer, patients can keep their pump up to date with the latest technology during its warranty period.



You know interfaces are important when...

Amazon, Apple, and Google unveil smart home collaboration

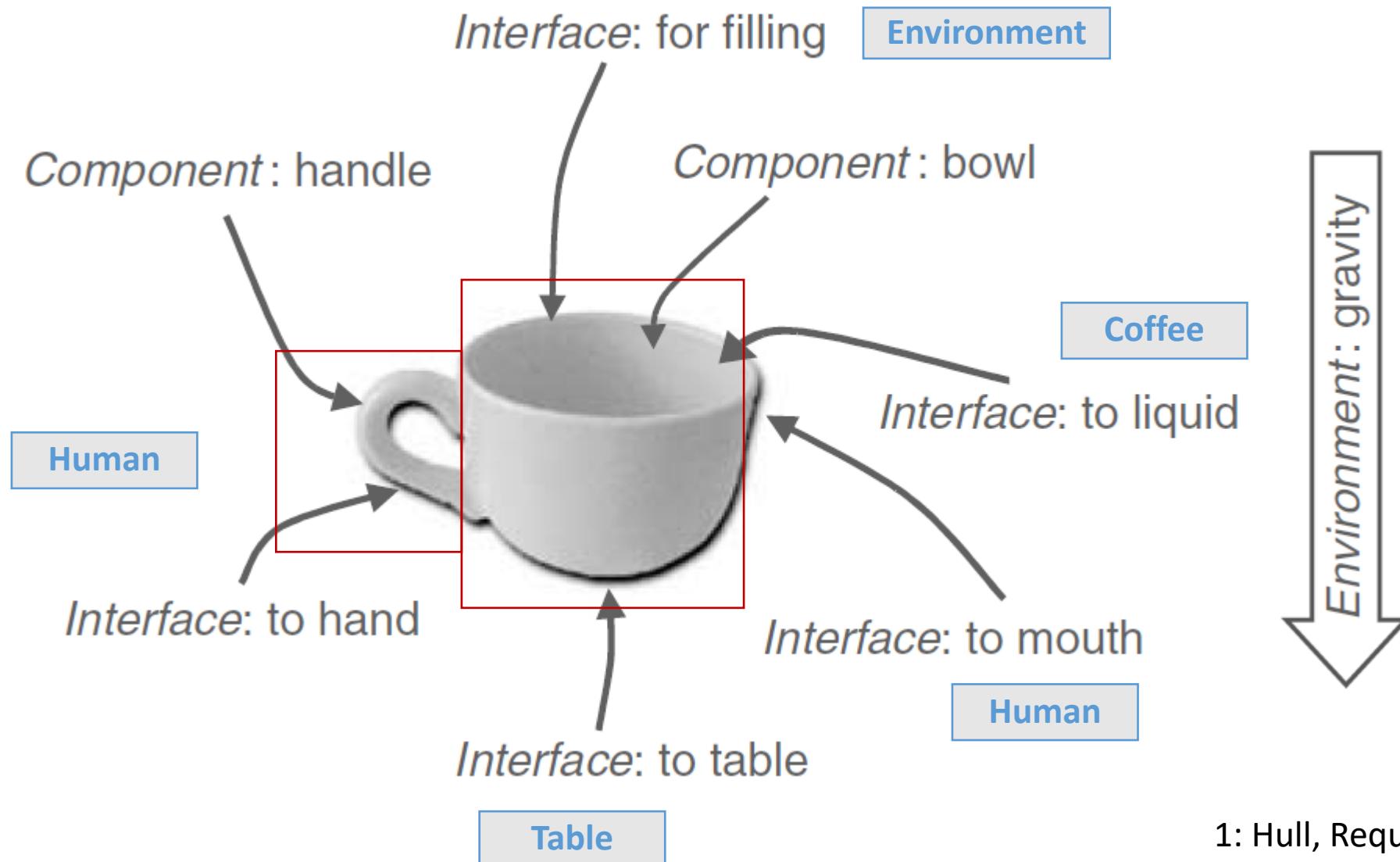


By [Pablo Valerio](#)

Posted on December 31, 2019



A Cup Viewed as a System¹



1: Hull, Requirements Engineering

- **Interface Requirements Document (IRD):** Defines the functional, performance, electrical, environmental, human, and physical requirements and constraints that exist at a common boundary between two or more functions or system elements
- **Interface Control Document or Interface Control Drawing (ICD):** Details the physical interface between two system elements, including the number and types of connectors, electrical parameters, mechanical properties, and environmental constraints.
- **Interface Definition Document (IDD) :** A unilateral document controlled by the end item provider. Provides the details of the interface for a design solution that is already established.

Validate Interfaces and Events

- For standard interfaces, ensure that the exact version of the relevant standard is identified
- For custom interfaces, agree all details of each interface with the owner of the other system
- Each event should be handled by when-requirements or their equivalent condition-action tables
- List of alternative conditions in condition-action tables cover all the possibilities exhaustively
- Check that each scenario referenced in the event handling requirement is defined as a use case
- Check that each interface is defined in the data dictionary

Group Activity

Using the system, you selected last class, identify four interfaces, 2 capabilities, and 2 scenarios

Be prepared to share with the class



WHEREWELEARN

Demo Button Demo Button

Demo Button Demo Button Demo Button

Demo Button Demo Button

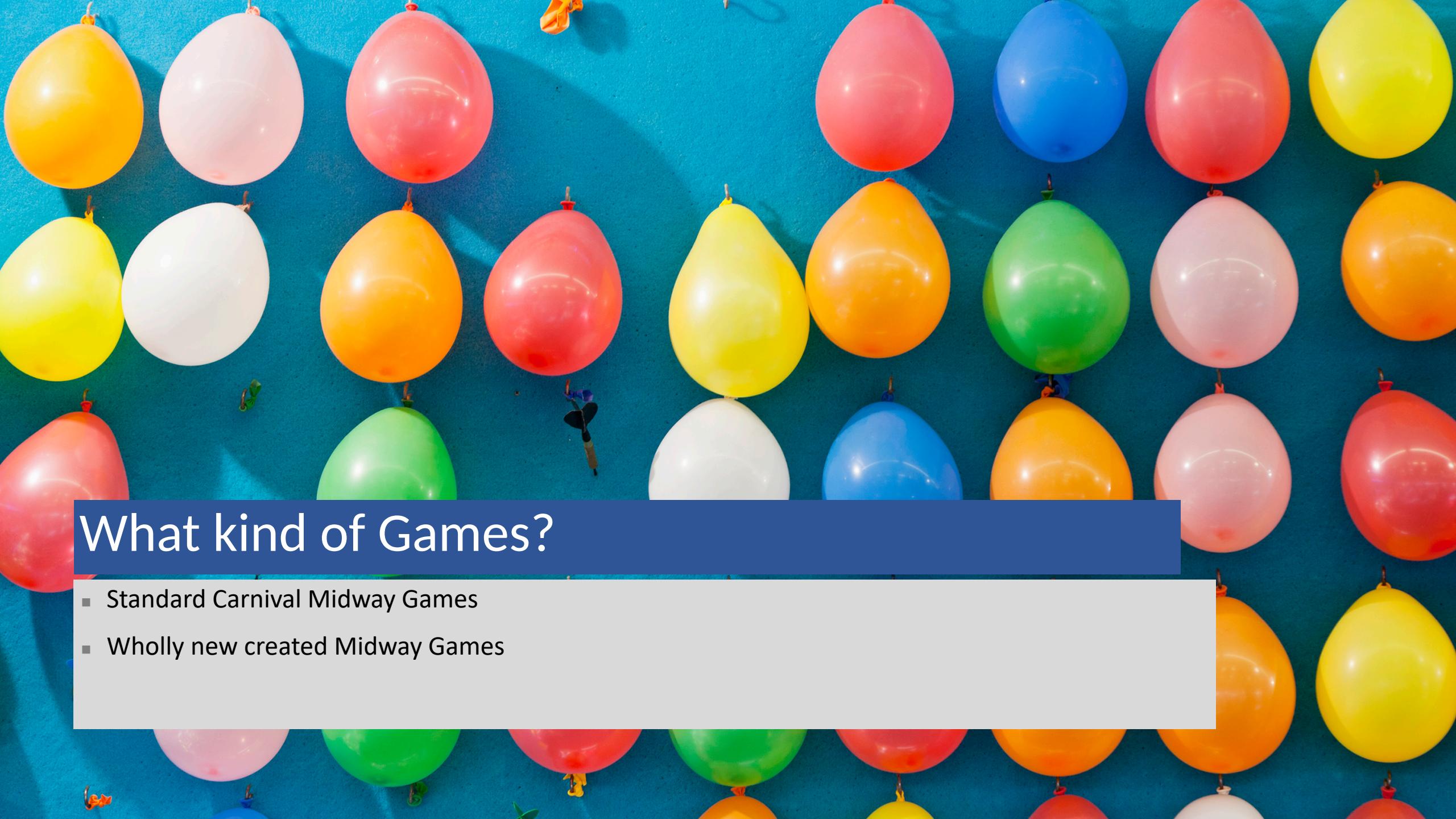
MASTER SYSTEMS DISPLAY



Objective

Teams of will produce carnival games to be played by members of the public at the CAMID Mini-Maker Faire.





What kind of Games?

- Standard Carnival Midway Games
- Wholly new created Midway Games

CLASS MINI PROJECT #1

- Identify a (GAME) that can be constructed with the use of an ARDUINO or Raspberry Pi device. (EG next page)
- Perform a series of discussions and interviews to identify project goals (during lecture time)
- Define project scope (1 slide)
- Create a context diagram (1slide)
- Describe 2 scenarios (up to 2 slides)
- The GAME must include a Software Component (Describe the software component)

EVERY TEAM MUST HAVE AN APPROVED PROJECT BY NEXT TUESDAY 5PM
EACH TEAM MUST PRESENT THEIR SYSTEM PROPOSAL AND SCOPE (THIS CLASS OR NEXT)
CLASSMATES AND GUESTS WILL VOTE FOR PROJECT ACCEPTANCE

Class Mini Project 1 Continued

- Each team will pitch their proposed system scope to the stakeholders next Tuesday
- Class and Invited Stakeholders will vote for funding.
- Once funded the team will be given certain resources to complete their project
- Each time will be assigned an external Stakeholder who will act as the client for the project
- Project competition deadlines will be announced in class as the projects progress.

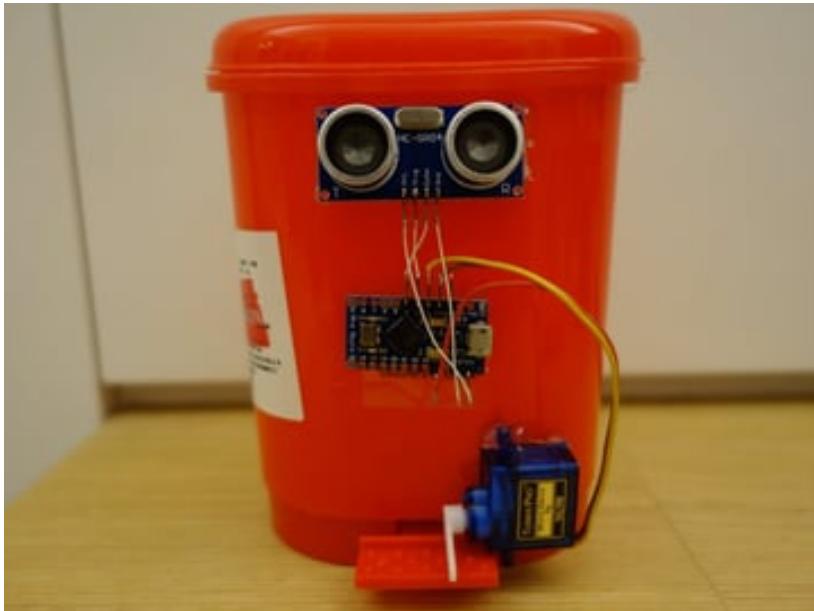


Clients will evaluate projects in terms of:

- Game Design
- Rules
- Playability
- Construction
- Appearance
- Fun



Touchless Trash Can



[\(Source: Arduino Project Hub\)](#)

Sensor Station



[\(Source: hackster.io\)](#)

Jar Dice Roller



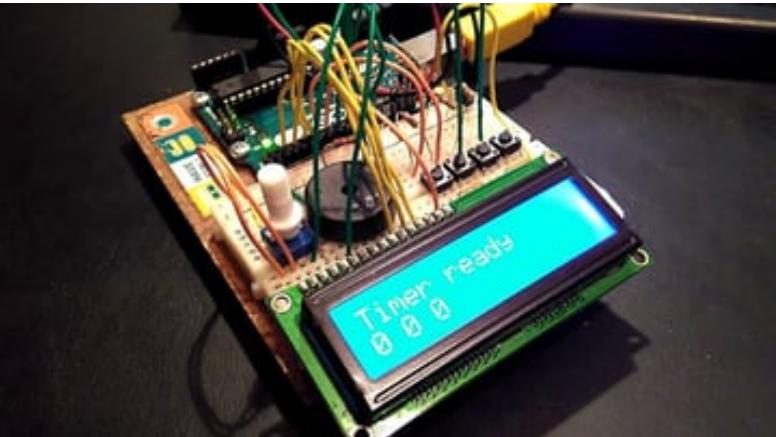
[\(Source: Instructables\)](#)

Chess Clock



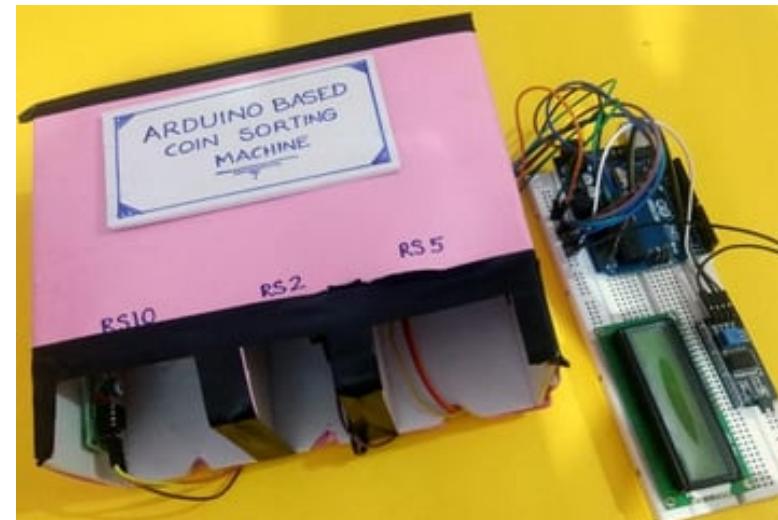
[\(Source: hackster.io\)](#)

Kitchen Timer



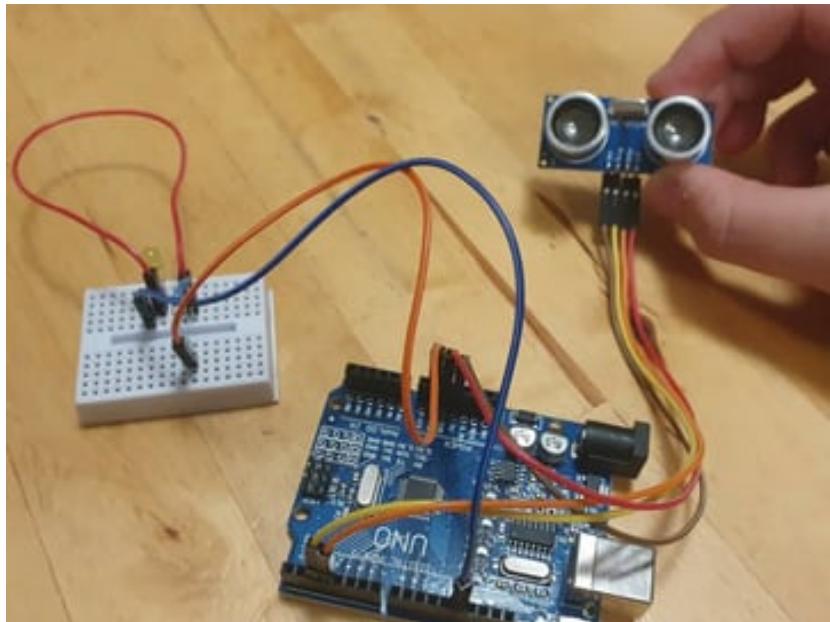
(Source: hackster.io)

Coin Sorting Machine



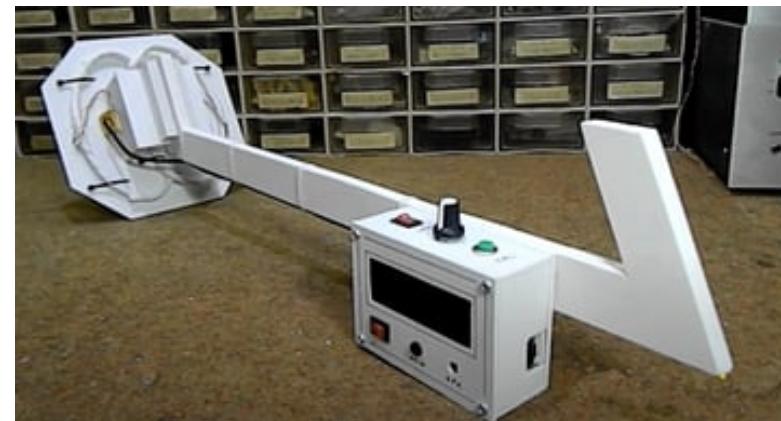
(Source: [Circuit Digest](https://circuitdigest.com))

Social Distancing Sensor



(Source: [hackster.io](https://www.hackster.io))

Metal Detector



(Source: [Arduino Project Hub](https://www.arduino-project-hub.com))

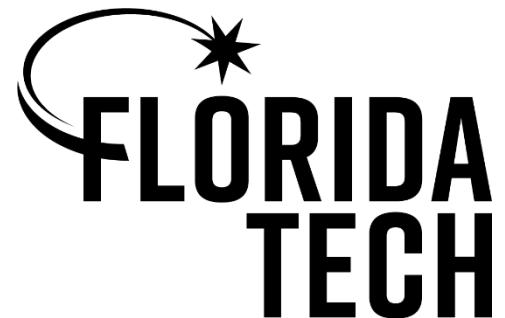
SYS 5460: Use Cases, Scenarios , and Functional Requirements

Contents:

- Use Cases
- Scenarios
- Modeling Recommendations
- Class Mini Project

Department of Computer & Engineering Sciences

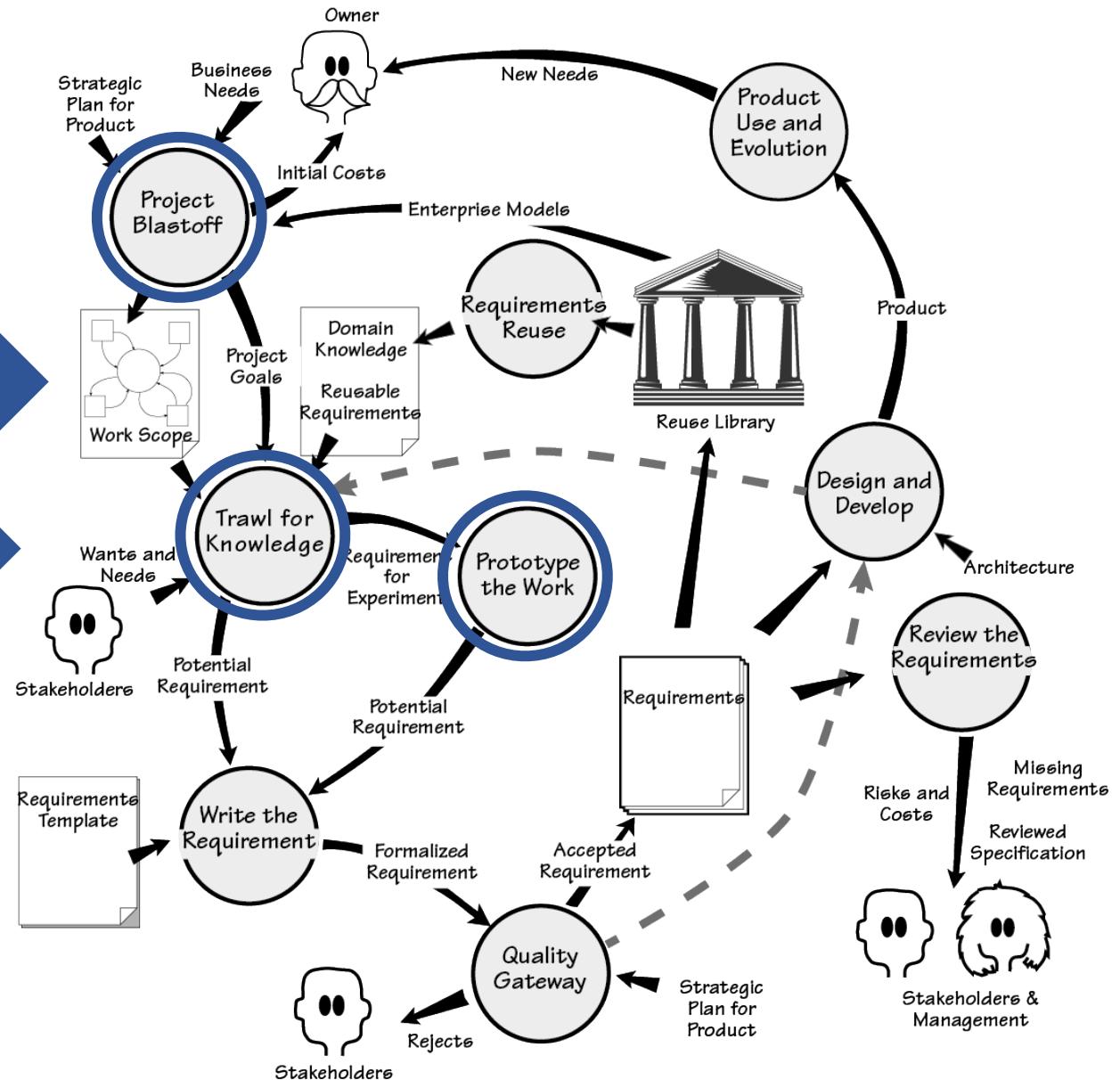
College of Engineering
Florida Institute of Technology



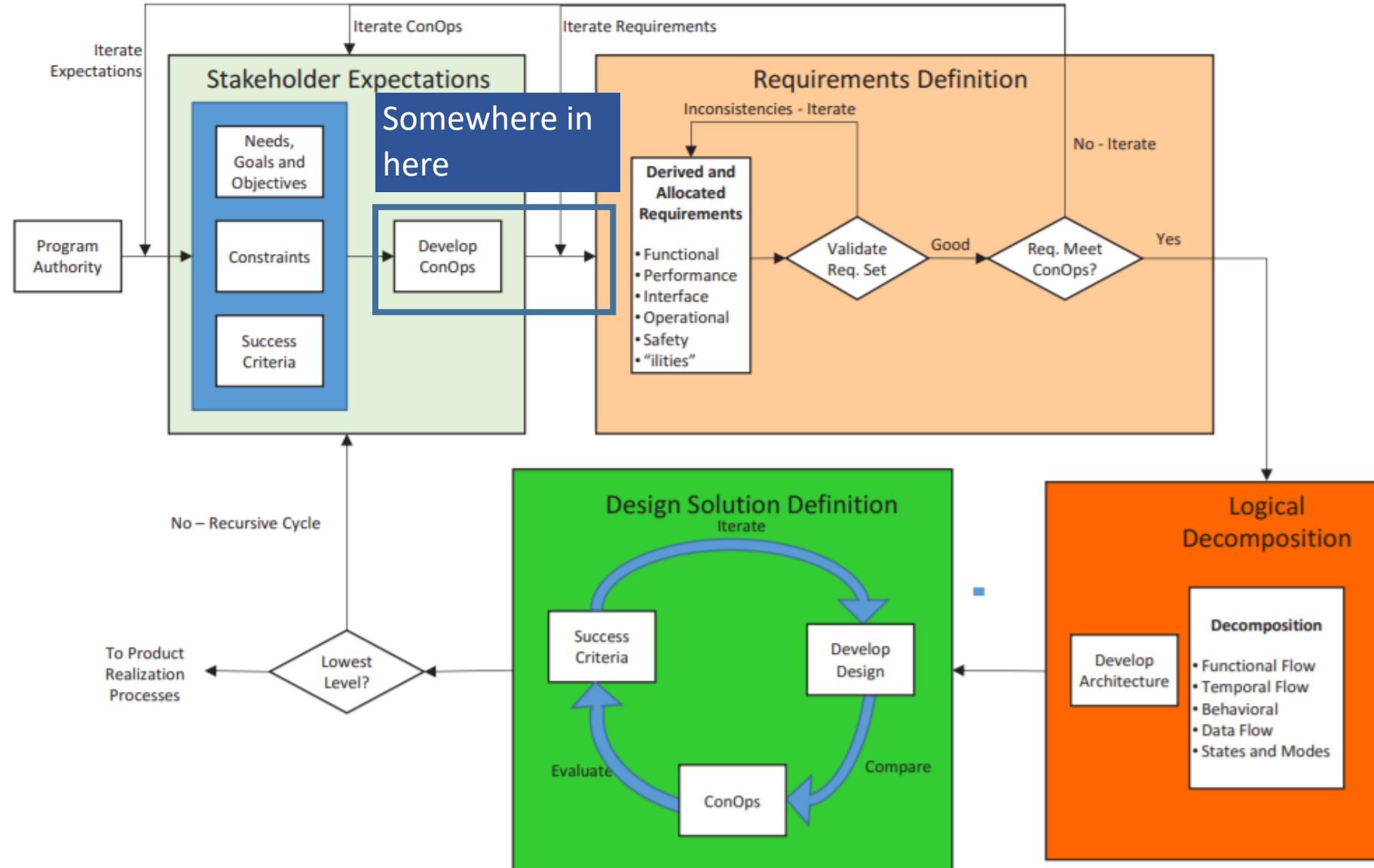
Volere Requirements Process (Robertson)

Model system functionality

Concept of Operations



Systems Engineering Process NASA Rev 2



Concept of Operations Document (IEEE 1362-1998)

Title page

Revision chart

Preface

Table of contents

List of figures

List of tables

1. Scope

 1.1 Identification

 1.2 Document overview

 1.3 System overview

2. Referenced documents

3. Current system or situation

 3.1 Background, objectives, and scope

 3.2 Operational policies and constraints

 3.3 Description of the current system or situation

 3.4 Modes of operation for the current system or situation

 3.5 User classes and other involved personnel

 3.6 Support environment

4. Justification for and nature of changes

 4.1 Justification of changes

 4.2 Description of desired changes

 4.3 Priorities among changes

 4.4 Changes considered but not included

5. Concepts for the proposed system

 5.1 Background, objectives, and scope

 5.2 Operational policies and constraints

 5.3 Description of the proposed system

 5.4 Modes of operation

 5.5 User classes and other involved personnel

 5.6 Support environment

6. Operational scenarios

7. Summary of impacts

 7.1 Operational impacts

 7.2 Organizational impacts

 7.3 Impacts during development

8. Analysis of the proposed system

 8.1 Summary of improvements

 8.2 Disadvantages and limitations

 8.3 Alternatives and trade-offs considered

9. Notes

Appendices

Glossary

Scenario

A **scenario** is a **scene** that illustrates some interaction with a proposed system.

A **scenario** is a tool used during requirements analysis to describe a specific use of a proposed system. Scenarios capture the system, as viewed from the outside, e.g., by a user, using specific examples.

Note on terminology

Some authors restrict the word "scenario" to refer to a user's total interaction with the system.

Other authors use the word "scenario" to refer to parts of the interaction.

In this course, the term is used with both meanings.

Describing a Scenario

Some organizations have complex documentation standards for describing a scenario.

At the very least, the description should include:

- A statement of the **purpose** of the scenario
- The individual **user** or **transaction** that is being followed through the scenario
- Assumptions about **equipment** or so(ware)
- The **steps** of the scenario

Developing a Scenario with a Client

Example of how to develop a scenario with a client

The requirements are being developed for a system that will enable university students to take exams online from their own rooms using a web browser.

Create a scenario for how a typical student interacts with the system.

In the next few slides, the questions in blue are typical of the questions to ask the client while developing the scenario.

Developing a Scenario with a Client: a Typical Student

Purpose: Scenario that describes the use of an online Exam system by a representative student

Individual: *[Who is a typical student?]* Student A, senior at FIT, major in computer science. *[Where can the student be located? Do other universities differ?]*

Equipment: Any computer with a supported browser. *[Is there a list of supported browsers? Are there any network restrictions?]*

Scenario:

1. Student A authenticates. *[How does a Florida Tech student authenticate?]*
2. Student A starts browser and types URL of Exam system. *[How does the student know the URL?]*
3. Exam system displays list of options. *[Is the list tailored to the individual user?]*

4. Student A selects CS 1234 Exam 1.
5. A list of questions is displayed, each marked to indicate whether completed or not. *[Can the questions be answered in any order?]*
6. Student A selects a question and chooses whether to submit a new answer or edit a previous answer. *[Is it always possible to edit a previous answer? Are there other options?]*
7. *[What types of question are there: text, multiple choice, etc.?]* The first question requires a written answer. Student A is submitting a new answer. The student has a choice whether to type the solution into the browser or to attach a separate file. Student A decides to attach a file. *[What types of file are accepted?]*

Developing a Scenario with a Client (continued)

8. For the second question, the student chooses to edit a previous answer. Student A chooses to delete a solution previously typed into the browser, and to replace it with an attached file. *[Can the student edit a previous answer, or must it always be replaced with a new answer?]*
9. As an alternative to completing the entire exam in a single session, Student A decides to saves the completed questions work to continue later. *[Is this always permitted?]*
10. Student A logs off.
11. Later Student A log in, finishes the exam, submits the answers, and logs out. *[Is this process any different from the initial work on this exam?]*
12. The Student A has now completed the exam. The student selects an option that submits the exam to the grading system. *[What if the student has not attempted every question? Is the grader notified?]*

Developing a Scenario with a Client (continued)

13. Student A now wishes to change a solution. The system does not permit changes once the solution has been submitted. *[Can the student still see the solutions?]*
14. Later Student A logs in to check the grades. *[When are grades made available? How does the student know?]*
15. Student A requests a regrade. *[What are the policies? What are the procedures?]*

Developing a Scenario with a Client (continued)

- Developing a scenario with a client clarifies many **functional requirements** that must be agreed before a system can be built, e.g., policies, procedures, etc.
- The scenario will (clarify the **requirements** for the **user interface**, but the design of the user interface should not be part of the scenario.

Although this scenario is quite simple, many details have been laid out.

Scenarios for Analyzing Special Requirements

Scenarios are very useful for analyzing special requirements.

Examples

- **Reversals.** In a financial system, a transaction is credited to the wrong account. What sequence of steps are used to reverse the transaction?
- **Errors.** A mail order company has several copies of its inventory database. What happens if they become inconsistent?
- **Malfeasance.** In a voting system, a voter has houses in two cities. What happens if he attempts to vote in both of them?

Scenarios for error recovery

Murphy's Law: "*If anything can go wrong, it will*". Create a scenario for everything that can go wrong and how the system is expected to handle it.

Models

Scenarios are useful in discussing a proposed system with a client, but requirements need to be made more precise before a system is fully understood.

This is the purpose of requirements **modeling**.

A **use case** provides such a model.

Describing a Use Case

Some organizations have complex documentation standards for describing a use case.

At the very least, the description should include:

- The **name** of the use case, which should summarize its purpose
- The **actor** or **actors**
- The **flow of events**
- Assumptions about **entry conditions**

Outline of Take Exam Use Case

Name of Use Case: Take Exam

Actor(s): ExamTaker

Flow of events:

1. ExamTaker connects to the Exam server.
2. Exam server checks whether ExamTaker is already authenticated and runs authentication process if necessary.
3. ExamTaker selects a exam from a list of options.
4. ExamTaker repeatedly selects a question and either types in a solution, attaches a file with a solution, edits a solution or attaches a replacement file.

Outline Specification of Use Case (continued)

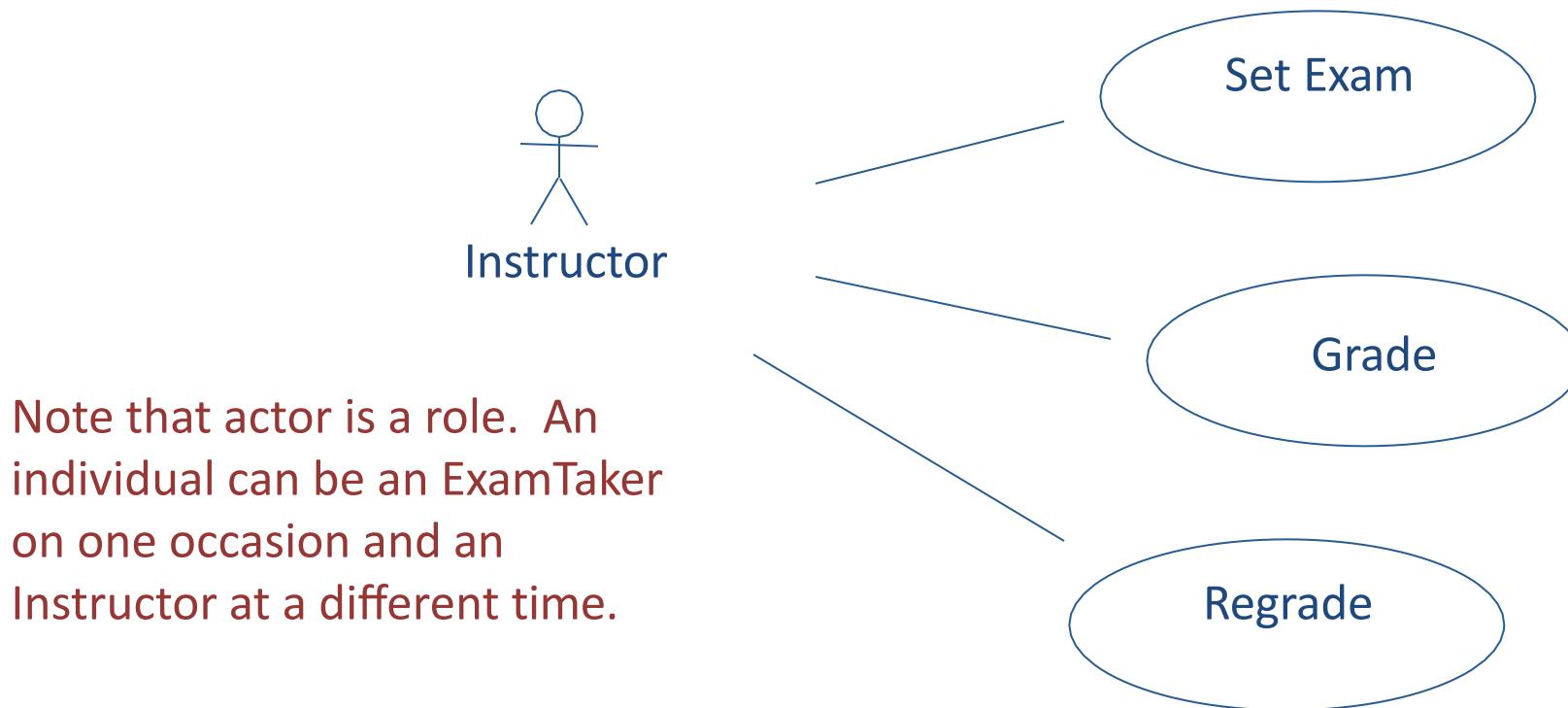
Flow of events (continued):

5. ExamTaker either submits completed exam or saves current state.
6. When a completed exam is submitted, Exam server checks that all questions have been attempted and either sends acknowledgement to ExamTaker, or saves current state and notifies ExamTaker of incomplete submission.
7. ExamTaker logs out.

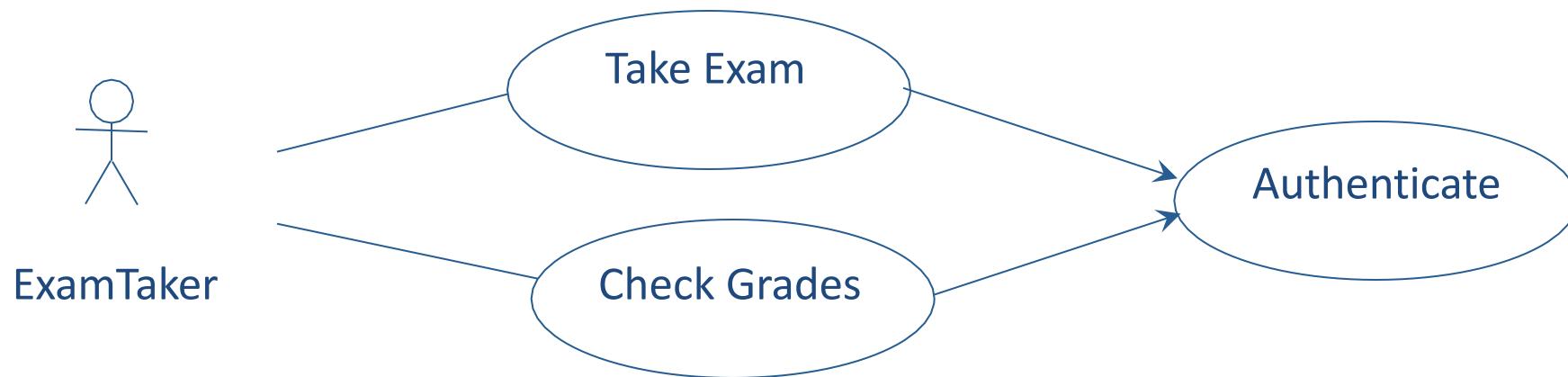
Entry conditions:

1. ExamTaker must have authentication credentials.
2. Computing requirements: supported browser.

Use Cases for Exam System (continued)



Relationships Between Use Cases:



The Authenticate use case may be used in other contexts

Scenarios and Use Cases in the Development Cycle

- Scenarios and use cases are both intuitive –easy to discuss with clients
- Scenarios are a tool for requirements analysis.
 - They are useful to validate use cases and in checking the design of a system.
 - They can be used as test cases for acceptance testing.
- Use cases are a tool for modeling requirements.
 - A set of use cases can provide a framework for the requirements specification.
 - Use cases are the basis for system and program design, but are often hard to translate into class models

Why do we do scenarios?

- To communicate a situation as it evolves through time in a series of steps
- Scenarios communicate requirements very effectively
- When scenarios describe interactions of humans and products they refer to an interface

What Scenarios to Write?

- Stakeholders' comments in interviews
- Goals in workshops
- known scenarios, e.g. problems with existing systems
- Study of the system context and scope
- Standard scenario patterns
 - Day in the Life of (DILO)
- Generally, the right level of scenario to start with is **a complete, end-to-end story** that deals with how a human operator uses a product to get results.
 - For instance, in an online bookstore, 'buy a book' is a complete story, whereas 'add item to cart' is a minor detail.
- It is also generally best to start with **normal, 'happy day'** scenarios

Contexts for Discovering Scenarios (Alexander)

- Interviews
- Observation
- Workshops

| Context | Advantages | Disadvantages | Techniques needed |
|------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| Interviews (Chapter 11) | Full attention on one person's story | Likelihood of hearing only part of a story or business process; need to piece together evidence from different interviews | More or less open-ended questioning of the interviewee; stimulation with existing documents |
| Observation (Chapter 11) | Direct experience builds understanding; can discover things not easily explained | Many important scenarios are rare, and unlikely to be observed | Note taking; recording and photography with permission; apprenticing; subject gives commentary while performing a task |
| Workshops (Chapter 12) | Participants stimulate each other, and fill in gaps in each other's knowledge as well as yours, as you reach different people's areas of expertise; ability to identify and resolve conflicts directly; ability to obtain group consensus on requirements directly | Cost of having several people focusing on the same activity; need for skilled facilitation | Facilitation using a range of techniques, e.g. role/action list, searching for negative scenarios, etc |

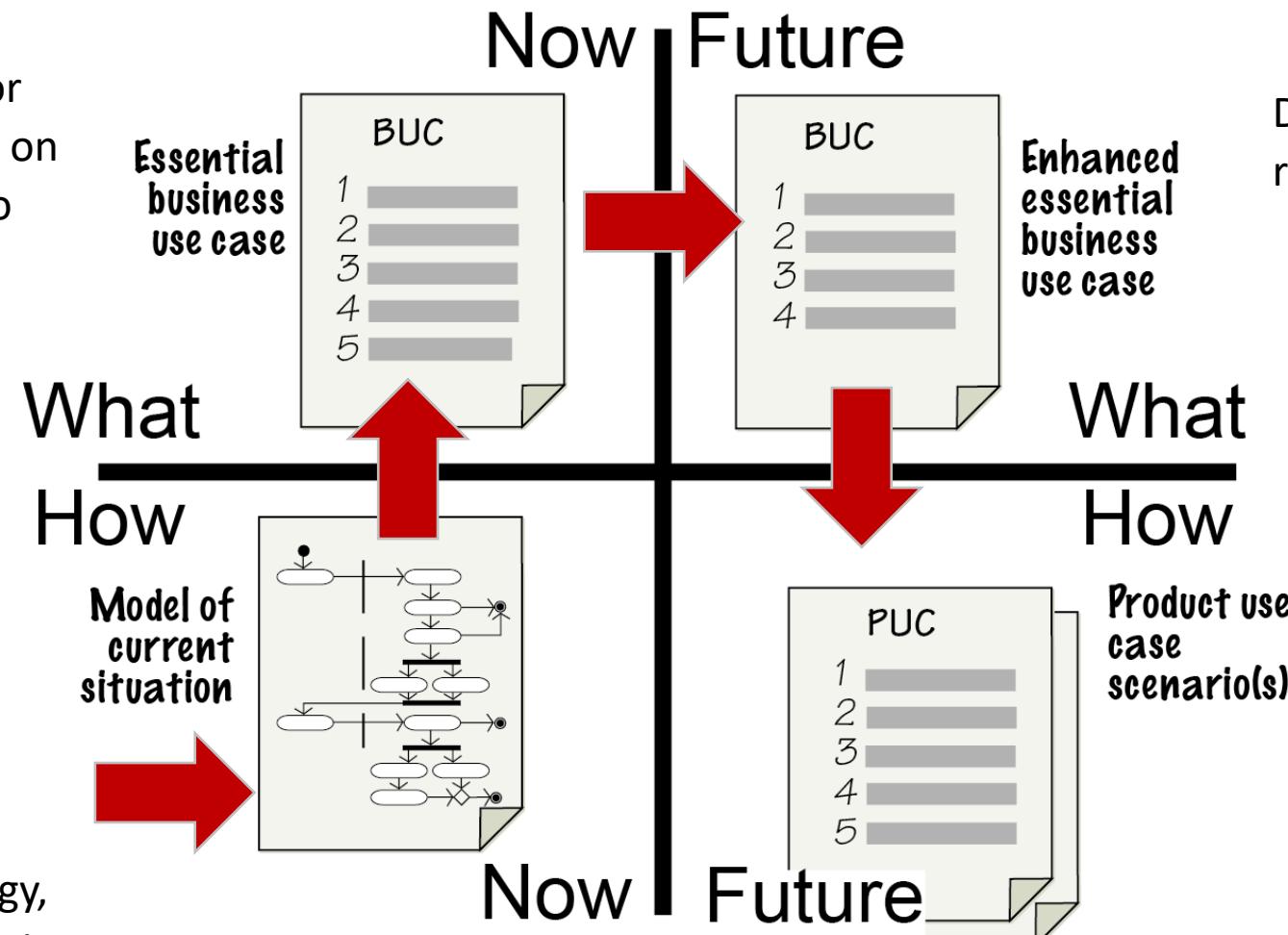
Interviewing Stakeholders

- Set the interview in context
- Limit the duration of the interview
- Have business use cases/diagrams as anchor for the interview
- Ask questions , listen to the answer and **feed back your understanding**
- Draw models and encourage the stakeholder to change them
- Use stakeholders terminology and artifacts
- Keep sample or copies of artifacts and log them for future reference
- Thanks stakeholders for their time and tell them what you learned and why it was valuable

Where are we going? (brown cow model...)

Remove implementation or tech references and focus on what the system should do

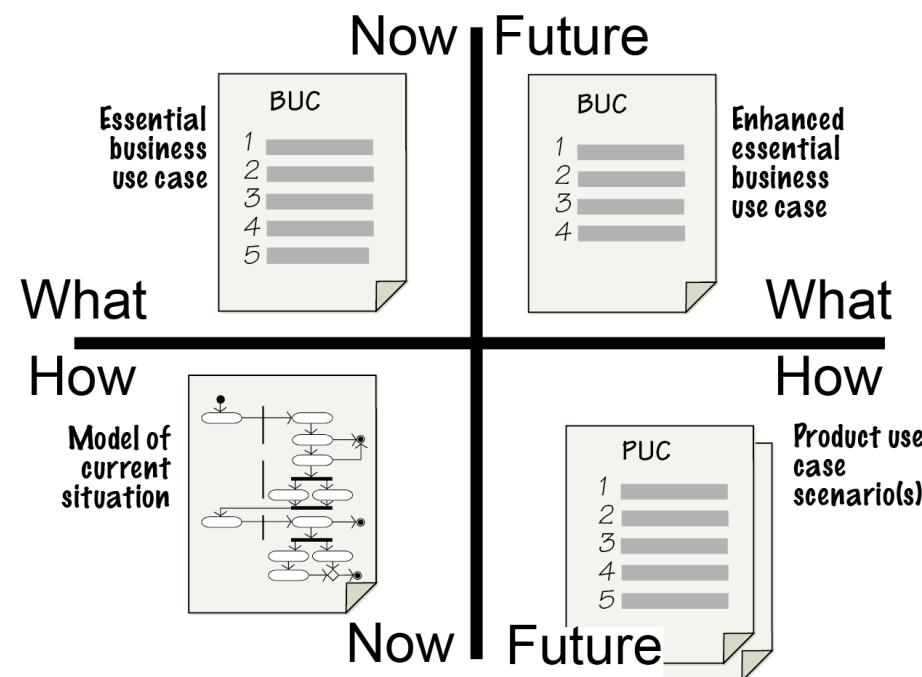
Desired system without reference to implementation



https://en.wikipedia.org/wiki/How_now_brown_cow

Use the How-Now to How-Future to Guide the Information Gathering

- Ask for What , How, When, Where, Why, Who to encourage description as opposed to Y/N
- Start with questions on the current situation. Ask for artifacts and methods.
- Then transition to what future “Are there other facts about<ti that would be useful to you”



Scenario Workshops

- Making role/action lists
- Making operator actions/machine actions lists
- Asking about standard patterns
- Acting scenes

Things to Avoid

1. Making design decision without noticing
2. Model swim lanes in real time (it make take to point #1)
 1. Step back and provide a curated list at the end of the workshop

Summary of

- Start with happy day scenarios
- Capture each story as simple as possible
- Take into consideration scope:
 - Normal transaction
 - DILO
 - Maintenance, retirement
- Move to special cases
 - Exceptions detection and handling

Negative Scenarios

- After the normal flow we need to test for things that should not happen (goal)
 - Exceptions: Events that interrupt normal progress
 - Intentional threats: Actions of hostile stakeholders which may cause exceptions
 - Unwanted scenarios: Undesirable combination of correct (allowed) behaviors
- Discovering negative scenarios
 - List events/triggers at the system interface
 - Define scope by agreeing upon all the events that will be covered by the product
 - Normal scenarios are complete when all the events are covered
 - List and agree on exceptions on the normal scenarios
 - Then what?
- Exception handling scenarios shall be simple, unambiguous and short
- Ignoring an exception is equivalent to accept a risk of failure
- For hardware components physical exceptions become important

Tips for Documenting Scenarios

- Decide what style will be most appropriate (colorful stories or something more analytic)
- Tell your story simply and directly
- Supply enough context to enable readers to see where the scenario fits in

Scenarios = Acceptance Test Cases

In general, system tests of products are not sufficient. They need to be supplemented by thorough testing of realistic scenarios. These should include scenarios that stress the whole system (product, people and external interfaces). Such scenarios often overload systems in unexpected ways.

Test engineers are skilled at devising troublesome scenarios of this kind. Classics include:

- **Peak load scenario (= stress test):** run the system up to (say) 110% capacity briefly, and see if it falls over.
- **Continuous load scenario (= soak test):** run the system at 100% capacity for N hours.
- **Failure injection scenario:** make something fail (pull a card out of a computer, reboot a server, unplug a network cable, etc) and see if the rest of the system can keep going.
- **Day in the life of (DILO) scenario:** play right through the whole of a normal day of operations, including logging on, taking meal breaks and so on.

Note that all the scenarios you discover – positive and negative – are likely candidates for test cases. The earlier you talk these through with your test people, the better.

CLASS MINI PROJECT #1

- Identify a Problem that can be resolved with the use of an ARDUINO device. (EG next page)
- Perform a series of discussions and interviews to identify project goals (todays lecture time)
- Define project scope (1 slide)
- Create a context diagram (1slide)
- How-Now to How-Future (1 slide)
- Describe 2 scenarios (up to 2 slides)

EVERY TEAM MUST HAVE AN APPROVED PROJECT BY NEXT TUESDAY 5PM
EACH TEAM MUST PRESENT THEIR SYSTEM PROPOSAL AND SCOPE (THIS CLASS OR NEXT)
CLASSMATES AND GUESTS WILL VOTE FOR PROJECT ACCEPTANCE

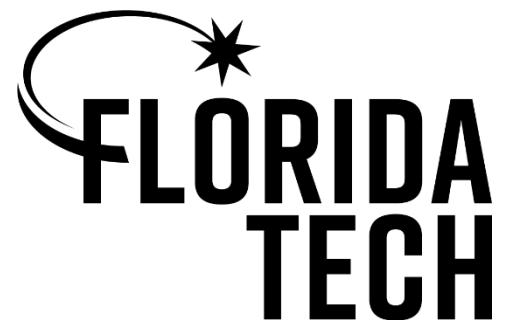
SYS 5460: Writing Requirements

Contents:

- Writing Requirements
- Requirement Characteristics
- Requirement Checks
- Examples

**Department of Computer &
Engineering Sciences**

College of Engineering
Florida Institute of Technology



ConOps is the bridge between customer and Project leader



How the customer explained it



How the project leader understood it



How the analyst designed it



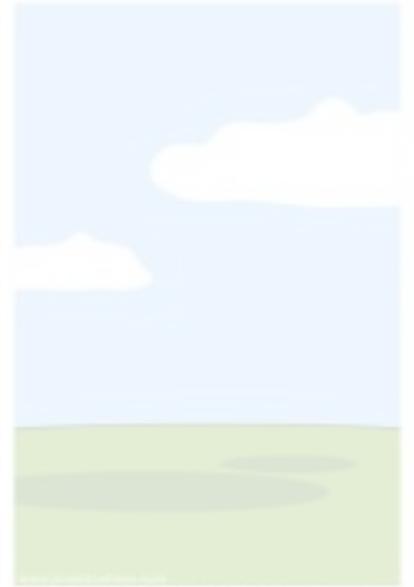
How the programmer wrote it



What the beta testers received



How the business consultant described it



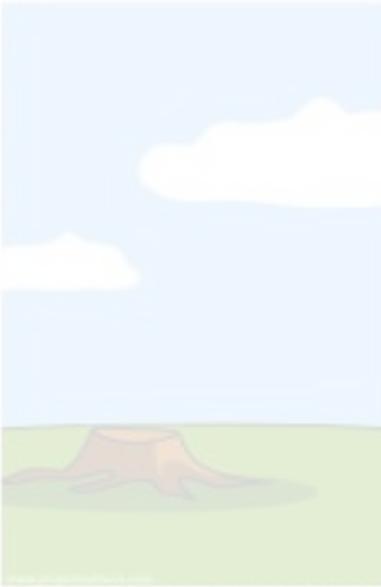
How the project was documented



What operations installed



How the customer was billed



How it was supported



iSwing

What marketing advertised



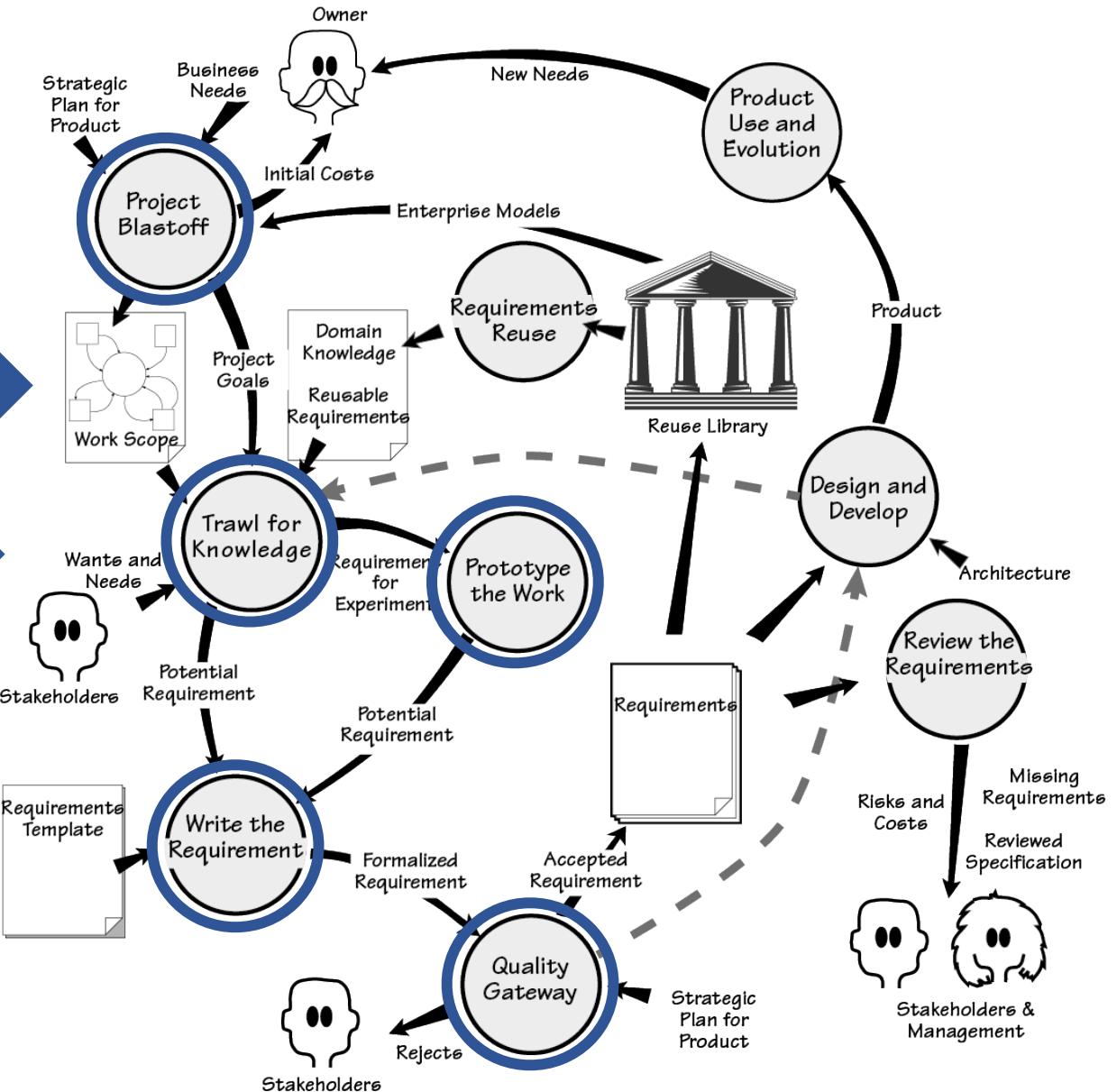
What the customer really needed

Volere Requirements Process (Robertson)

Model system functionality

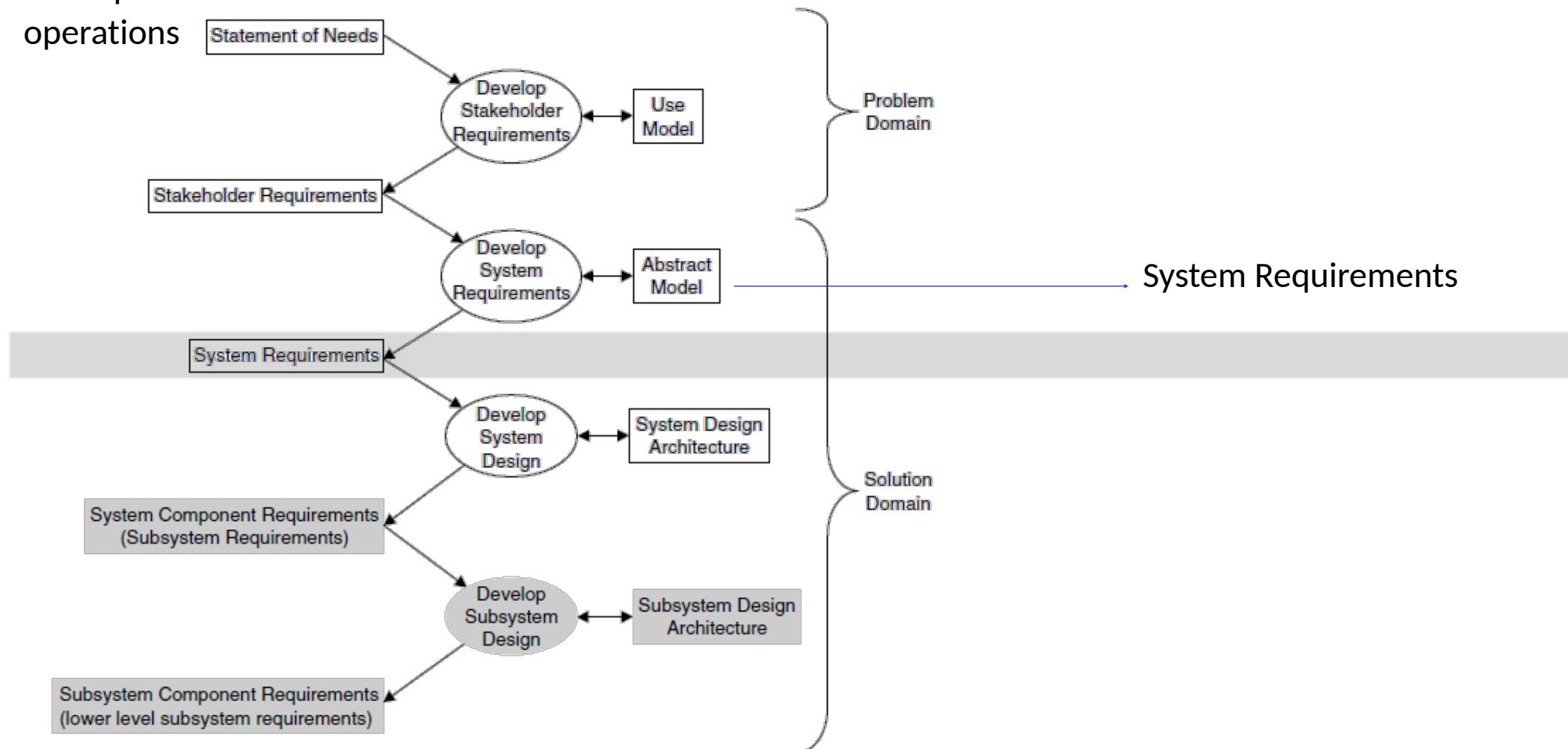
Concept of Operations

Functional Requirements



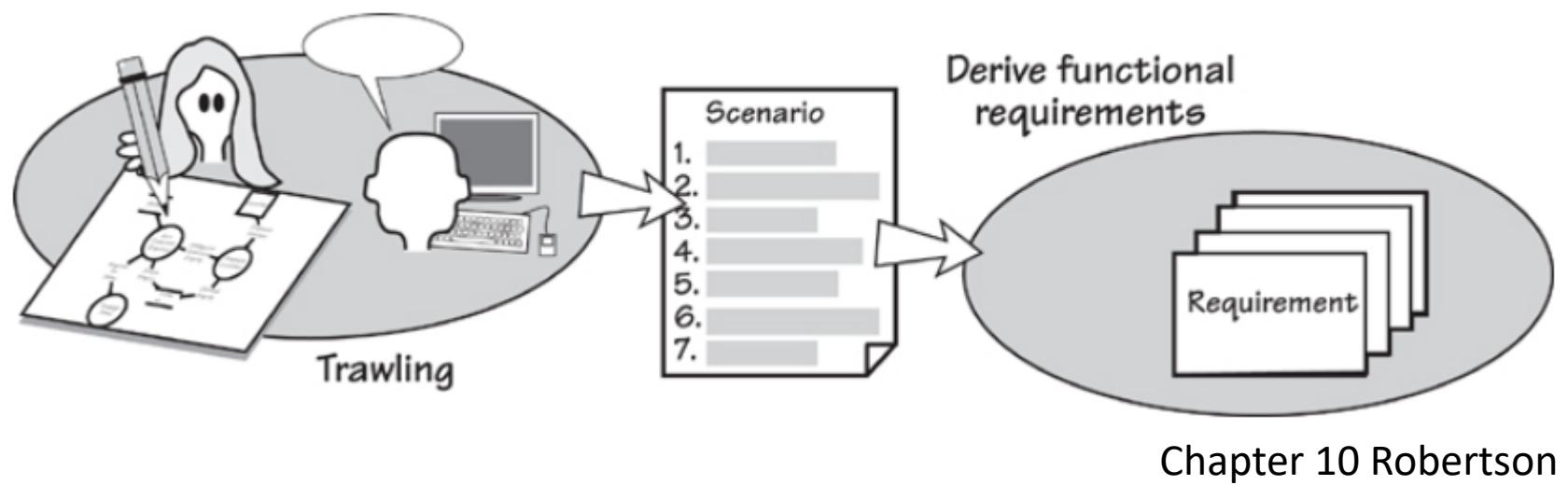
Stakeholders->Concept of Operations->Model

Concept of operations



Prerequisites

- System in the problem space is defined



Functional Requirement

- Functional requirements describe what the product has to do to support and enable the owner's work. They should be, as far as possible, independent of the technology used by the eventual product (Robertson ch 10)
- A statement that identifies what a product or process must accomplish to produce required behavior and/or results. 2. A requirement that specifies a function that a system or system component must be able to perform. (IEEE and ISO/IEC 2010, p. 153)
- Functional requirements have the following essential characteristics:
 - Define what the system should do
 - Be action oriented
 - Describe tasks or activities
 - Are associated with the transformation of inputs to outputs

Writing Functional Requirements

- Requirements are written as a single sentence with a single verb.
- It is common to use shall, must, will, might, could etc. to indicate the priority of the requirement. This may result in semantic confusion.
- Use one consistent form (shall) and use a separate attribute to indicate its priority
- Other practices include
 - Shall: Mandatory
 - Should: Optional
 - Will: Deferred compliance
- When in doubt IEEE
- For requirement modeling additional attributes with priority can be implemented

Requirements Style and Interpretation

- It is common to use shall, must, will, might, could etc. to indicate the priority of the requirement. This may result in semantic confusion.
- Use one consistent form (shall) and use a separate attribute to indicate its priority
- Other practices include
 - **Shall: Mandatory**
 - Should: Optional
 - Will: Deferred compliance
- When in doubt IEEE

Characteristics of Individual Requirements (SEBoK)

| Characteristic | Description |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Necessary | The requirement defines an essential capability, characteristic, constraint, and/or quality factor. If it is not included in the set of requirements, a deficiency in capability or characteristic will exist, which cannot be fulfilled by implementing other requirements. |
| Appropriate | The specific intent and amount of detail of the requirement is appropriate to the level of the entity to which it refers (level of abstraction). This includes avoiding unnecessary constraints on the architecture or design to help ensure implementation independence to the extent possible. |
| Unambiguous | The requirement is stated in such a way so that it can be interpreted in only one way. |
| Complete | The requirement sufficiently describes the necessary capability, characteristic, constraint, or quality factor to meet the entity need without needing other information to understand the requirement. |
| Singular | The requirement should state a single capability, characteristic, constraint, or quality factor. |
| Feasible | The requirement can be realized within entity constraints (e.g., cost, schedule, technical, legal, regulatory) with acceptable risk. |
| Verifiable | The requirement is structured and worded such that its realization can be proven (verified) to the customer's satisfaction at the level the requirements exists. |
| Correct | The requirement must be an accurate representation of the entity need from which it was transformed. |
| Conforming | The individual requirements should conform to an approved standard template and style for writing requirements, when applicable. |

Attributes of the Requirements Document

Ability

- Ability uniquely to identify every statement of requirement
- Ability to classify every statement of requirement in multiple ways, such as:
 - By importance
 - By type (e.g. functional, performance, constraint, safety)
 - By urgency (when it has to be provided)
- Ability to track the status of every statement of requirement, in support of multiple processes, such as:
 - Review status
 - Satisfaction status
 - Qualification status
- Ability to elaborate a requirement in multiple ways, such as by providing:
 - Performance information
 - Quantification
 - Test criteria
 - Rationale
 - Comments
- Ability to view a statement of requirement in the document context, i.e. alongside its surrounding statements
- Ability to navigate through a requirements document to find requirements according to a particular classification or context
- Ability to trace to any individual statement of requirement

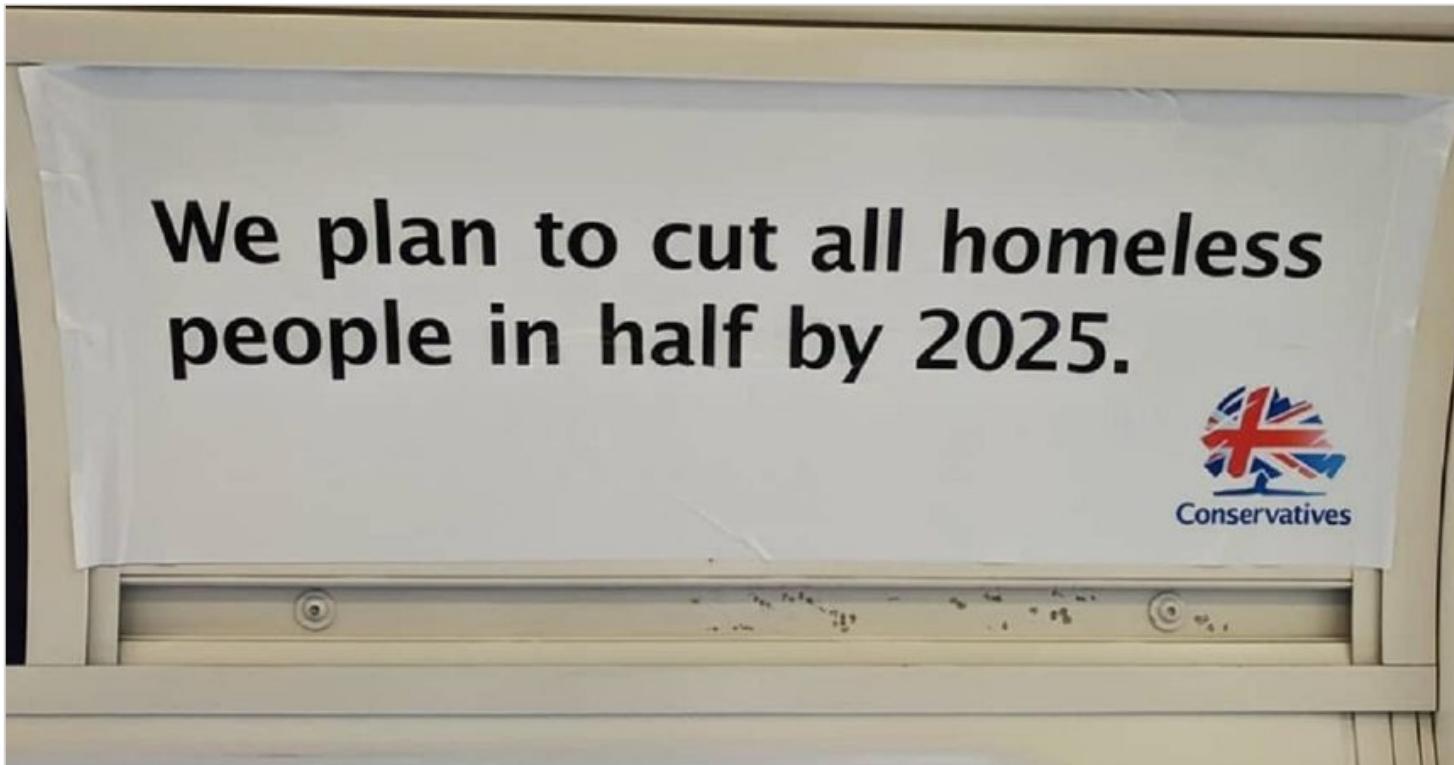


Helps to perform QA on requirements

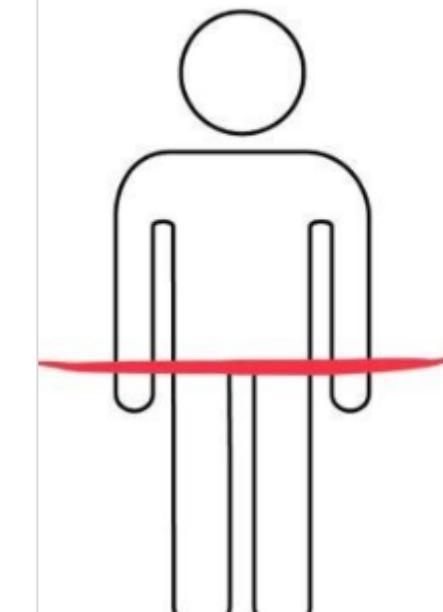


minimize the number of requirements
understand large amounts of information
find sets of requirements relating to particular topics
detect omissions and duplications
eliminate conflicts between requirements
manage iteration (e.g. delayed requirements)
reject poor requirements
evaluate requirements
reuse requirements across projects

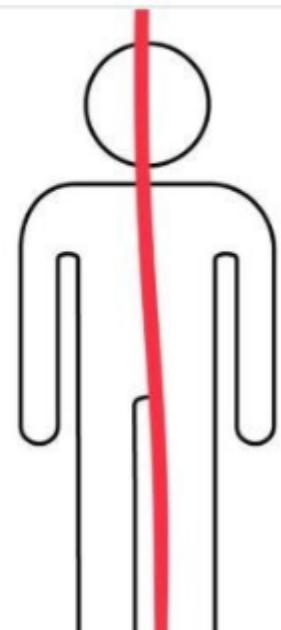
Ambiguity...



like this



or like this



Key objectives when writing requirements

- Make the requirements document readable
- Make the set of requirements processable

Characteristic of a Set of Requirements (

| Characteristic | Description |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Complete | The requirement set stands alone such that it sufficiently describes the necessary capabilities, characteristics, constraints, and/or quality factors to meet the entity needs without needing other information. In addition, the set does not contain any to be defined (TBD), to be specified (TBS), or to be resolved (TBR) clauses. |
| Consistent | The set of requirements contains individual requirements that are unique, do not conflict with or overlap with other requirements in the set, and the units and measurement systems they use are homogeneous. The language used within the set of requirements is consistent, i.e., the same word is used throughout the set to mean the same thing. |
| Feasible | The requirement set can be realized within entity constraints (e.g., cost, schedule, technical, legal, regulatory) with acceptable risk. (Note: Feasible includes the concept of "affordable".) |
| Comprehensible | The set of requirements must be written such that it is clear as to what is expected by the entity and its relation to the system of which it is a part. |
| Able to be validated | It must be able to be proven the requirement set will lead to the achievement of the entity needs within the constraints (such as cost, schedule, technical, legal and regulatory compliance). |

More on Requirements Language

- A typical English word has 10 synonyms, so even a 4 word requirement statement has 10,000 possible interpretations
- Bellagamba* suggest the following ambiguity check:
 - Compliance level
 - Completeness
 - Precision
 - Comprehension
 - Referencing
 - Vague words
 - Functional requirement
 - Acronyms
 - English unit usage
 - Word emphasis

*Bellagamba Larry, Systems Engineering and Architecting Ch4 , CRC Press 2012

Example of Requirement Check

- Compliance check (Bellagamba). Check for the following words:
 - **Type 1.** also, anticipate, apply, applies, are to, aspire, can, could, crave, demand, desire, expect, force, forcing, ideally, goal, got to, has to, is to, might, must, necessary, necessitates, need, needed, needs, obligate, obligation, require, prefer, preference, should, stipulate, want, wants, will, would
 - **Type 2.** can't, don't, mustn't, needn't, not, shouldn't, won't

Suggested Form

- ▶ For type 1:
 - If feature **M** is mandatory for entity **E**, use: *The E shall M*
 - If feature **O** is optional for entity **E**, use: *The E should O*
 - If feature **D** is deferred for entity **E**, use: *The E will D*
- ▶ For type 2:
 - Phrase as a statement of inclusion rather than exclusion

Example of Compliance Check (Cont'd)

- If entity E may exhibit feature V with values between $V1$ and $V2$, use:
 - *The E shall V between $V1$ and $V2$ with equal preference over the range.*

*Bellagamba Larry, Systems Engineering and Architecting Ch4 , CRC Press 2012

Completeness

- Completeness ambiguity is caused when missing information is included in the requirement statement.
 - not known, tbd, tbr, tbs, tbx, to be determined, to be provided, to be reviewed,to be specied, to be
 - Delete the requirement until complete, or provide the incomplete information supplied, unknown

Precision Ambiguity

- Precision ambiguity is uncertainty in how to interpret numerical information
 - Type 1: is above, at least, minimum of, no less than, not less than, not to be less than, exceed.
 - Type 2: is at most, below, maximum, no greater than, not greater than, not to be greater than, under, up to, within.
 - Type 3: is about, almost, approximately, at, between, close to, exactly, give or take, more or less, near, of, or so, plus or minus, roughly, tolerance, use, $+/-$.
 - Type 4: is average.
- For type 1 (feature F to be $\geq L$):
 - The probability F is greater than (or equal to) L shall be p
- For type 2 (feature F to be $\leq U$):
 - The probability F is less than (or equal to) U , shall be p
- For type 3 (feature F to be $C | \text{feature } F > (=) L \text{ and } F < (=) U$):
 - The F shall be C
 - The probability F is greater than (or equal) to L and less than (or equal) to U shall be p
- For average:
 - use mean, median, mode, and arithmetic or geometric, or explain how average is to be determined

Comprehension Ambiguity

- Type 1. Using more than two instances of *shall*, a semicolon, or a colon.
- Type 2. Using more than two instances of *and* or *or*.
- Type 3. Using more than one *but*.

- Sugestions:
- If (C1 then) the E1 (shall) be A1 (else if C2 then) the E2 (shall) be A2
 - Ci: conditions
 - Ei: entities
 - Ai: Attrbutes

- Preferences or goals are desired, non-mandatory, non-binding provisions and use 'should'
- Suggestions or allowances are non-mandatory, non-binding provisions and use 'may'
- Non-requirements, such as descriptive text, use verbs such as 'are', 'is', and 'was'. It is best to avoid using the term 'must', due to potential misinterpretation as a requirement
- Use positive statements and **avoid negative requirements** such as 'shall not'
- Use active voice: avoid using passive voice, such as 'shall be able to select'

- Complete. The set of requirements needs no further amplification because it contains everything pertinent to the definition of the system or system element being specified. In addition, the set contains no To Be Defined (TBD), To Be Specified (TBS), or To Be Resolved (TBR) clauses
- Consistent. The set of requirements does not have individual requirements which are contradictory. Requirements are not duplicated. The same term is used for the same item in all requirements
- Affordable. The complete set of requirements can be satisfied by a solution that is obtainable/feasible within life cycle constraints (e.g., cost, schedule, technical, legal, regulatory)
- Bounded. The set of requirements maintains the identified scope for the intended solution without increasing beyond what is needed to satisfy user needs

- Superlatives (such as 'best', 'most')
- Subjective language (such as 'user friendly', 'easy to use', 'cost effective')
- Vague pronouns (such as 'it', 'this', 'that')
- Ambiguous adverbs and adjectives (such as 'almost always', 'significant', 'minimal')
- Open-ended, non-verifiable terms (such as 'provide support', 'but not limited to', 'as a minimum')
- Comparative phrases (such as 'better than', 'higher quality')
- Loopholes (such as 'if possible', 'as appropriate', 'as applicable')
- Incomplete references (not specifying the reference with its date and version number; not specifying just the applicable parts of the reference to restrict verification work)
- Negative statements (such as statements of system capability not to be provided) All as

Examples

1. The assignment is due on March 2
2. An aircraft that is non-friendly and has an unknown mission or the potential to enter restricted airspace within 5 minutes shall raise an alert.
3. All lights shall have their switch
4. The trucks shall treat the roads before they freeze
5. The system shall shut off the pumps if the water level remains above 100 meters for more than 4 seconds
6. The product shall show the weather for the next 24 hours
7. The product shall communicate all roads predicted to freeze.

Discussion

Provide creative implementations to the requirement

Suggestions to avoid ambiguity

- Requirements elicitation and requirements documentation
 - Eliminate all pronouns from your requirements and replace them with the subject or object to which the pronouns refer
 - Read it aloud. If possible, have a colleague read it aloud
 - Confirm with your stakeholder that you both reach the same understanding of the requirement
- Keep in mind that you are writing a *description* of the requirement. The real requirement is revealed when you write the full documentation (fit criterion+ rationale)
- Establish a context, because language is interpreted always in context, and if this context is not made explicit and agreed to by all the stakeholders in an elicitation session, misinterpretations are likely

Requirement engineer's paraphrasing of customers' and users' statements in her own words is an effective way for the requirements engineer to get the customers and users to spot their own ambiguities

Suggestions to avoid ambiguity (Continued)

- Increase precision of natural language
- Provide contextual information (rationale)
- Provide context and data definitions
- Look for patterns of ambiguity in the existing documentation
- Compare (and share) interpretation from different stakeholders

Using Requirements Templates or Boiler Plates

- [when?] [under what conditions?] THE SYSTEM SHALL | SHOULD | WILL <process> <thing to be processed>[<process detail>*]

[Condition] [Subject] [Action] [Object] [Constraint]

EXAMPLE: When signal x is received [Condition], the system [Subject] shall set [Action] the signal x received bit [Object] within 2 seconds [Constraint].

Or

[Condition] [Action or Constraint] [Value]

EXAMPLE: At sea state 1 [Condition], the Radar System shall detect targets at ranges out to [Action or Constraint] 100 nautical miles [Value].

Or

[Subject] [Action] [Value]

EXAMPLE: The Invoice System [Subject], shall display pending customer invoices [Action] in ascending order [Value] in which invoices are to be paid.

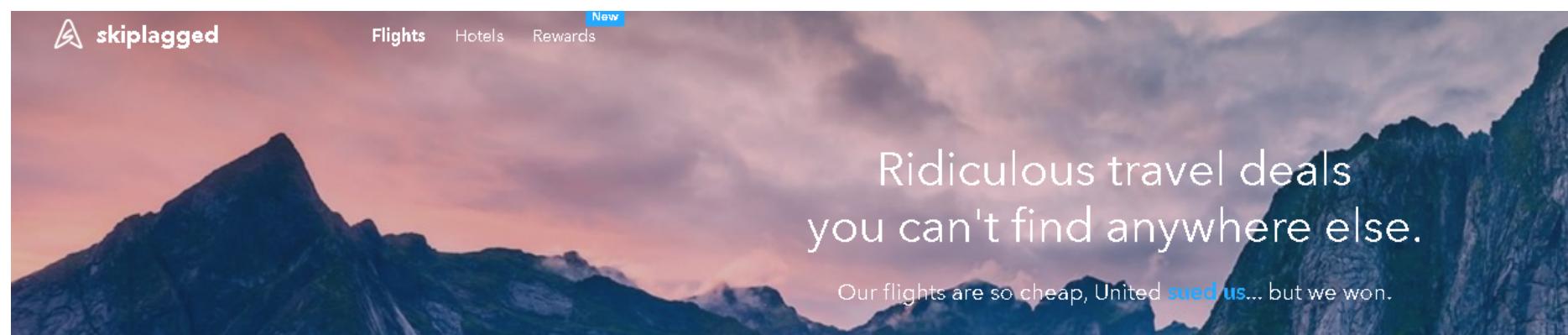
■ For Stakeholder capabilities

The <stakeholder type> shall be able to <capability> within <performance> of <event> while <operational condition>.

The weapons operator shall be able to fire a missile within 3 seconds of radar sighting while in severe sea conditions.

The <stakeholder> shall not be placed in breach of <applicable law>.

E.g. The ambulance driver shall not be placed in breach of national road regulations.



Find flights the airlines don't want you to see.
We're exposing **loopholes** in airfare pricing to save you money.

- For System capabilities

The **<system>** shall **<function>**
not less than **<quantity>** **<object>**
while **<operational condition>**.

*E.g. The communications system shall sustain telephone contact
with not less than 10 callers
while in the absence of external power.*

The **<system>** shall **<function>** **<object>**
every **<performance>** **<units>**.

*E.g. The coffee machine shall produce a hot drink
every 10 seconds.*

Requirement Expressions

■ Requirement Expression= Requirement statement + attributes

| Category | Example values |
|----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Identification</i> | |
| • Identifier | Unique reference |
| • Name | Unique name summarising the subject of the requirement |
| <i>Intrinsic characteristics</i> | |
| • Basic type | Functional, performance, quality factor, environment, interface, constraint, non-requirement |
| • Quality factor sub-type | Availability, flexibility, integrity, maintainability, portability, reliability, safety, security, supportability, sustainability, usability, workmanship |
| • Product/process type | Product, process, data, service |
| • Quantitative/qualitative type | Quantitative, qualitative |
| • Life-cycle phase | Pre-concept, concept, development, manufacturing, integration/test, deployment/delivery/installation, operation, support, disposal |
| <i>Priority and importance</i> | |
| • Priority (compliance level) | Key, mandatory, optional, desirable <i>or</i> Must, should, could, wish (MoSCoW) |
| • Importance | 1 to 10 |
| <i>Source and ownership</i> | |
| • Derivation type | Allocation, decomposition |
| • Source (origin) | Name of document or stakeholder |

| Category | Example values |
|------------------------------------|-----------------------------------------------------------------------------|
| • Owner | Name of stakeholder |
| • Approval authority | Name or person |
| <i>Context</i> | |
| • Requirements set/document | (Best handled through positioning the requirement in a structured document) |
| • Subject | |
| • Scope | |
| <i>Verification and validation</i> | |
| • V&V method | Analysis, inspection, system test, component test |
| • V&V stage | (See life-cycle phase) |
| • V&V status | Pending, pass, failed, inconclusive |
| • Satisfaction argument | Rationale for choice of decomposition |
| • Validation argument | Rationale for choice of V&V methods |
| <i>Process support</i> | |
| • Agreement status | Proposed, being assessed, agreed |
| • Qualification status | Not qualified, qualified, suspect |
| • Satisfaction status | Not satisfied, satisfied, suspect |
| • Review status | To be reviewed, accepted, rejected |
| <i>Elaboration</i> | |
| • Rationale | Textual statement about why the requirement is present |
| • Comments | Textual comments of clarification |
| • Questions | Questions to be posed for clarification |

Requirement Expressions (continued)

Elaboration

| | |
|-------------|--------------------------------------------------------|
| • Rationale | Textual statement about why the requirement is present |
| • Comments | Textual comments of clarification |
| • Questions | Questions to be posed for clarification |
| • Responses | Responses received for clarification |

Miscellaneous

| | |
|------------------------|-----------------------------------------------|
| • Maturity (stability) | Number of changes/time |
| • Risk level | High, medium, low |
| • Estimated cost | |
| • Actual cost | |
| • Product release | Version(s) of product meeting the requirement |

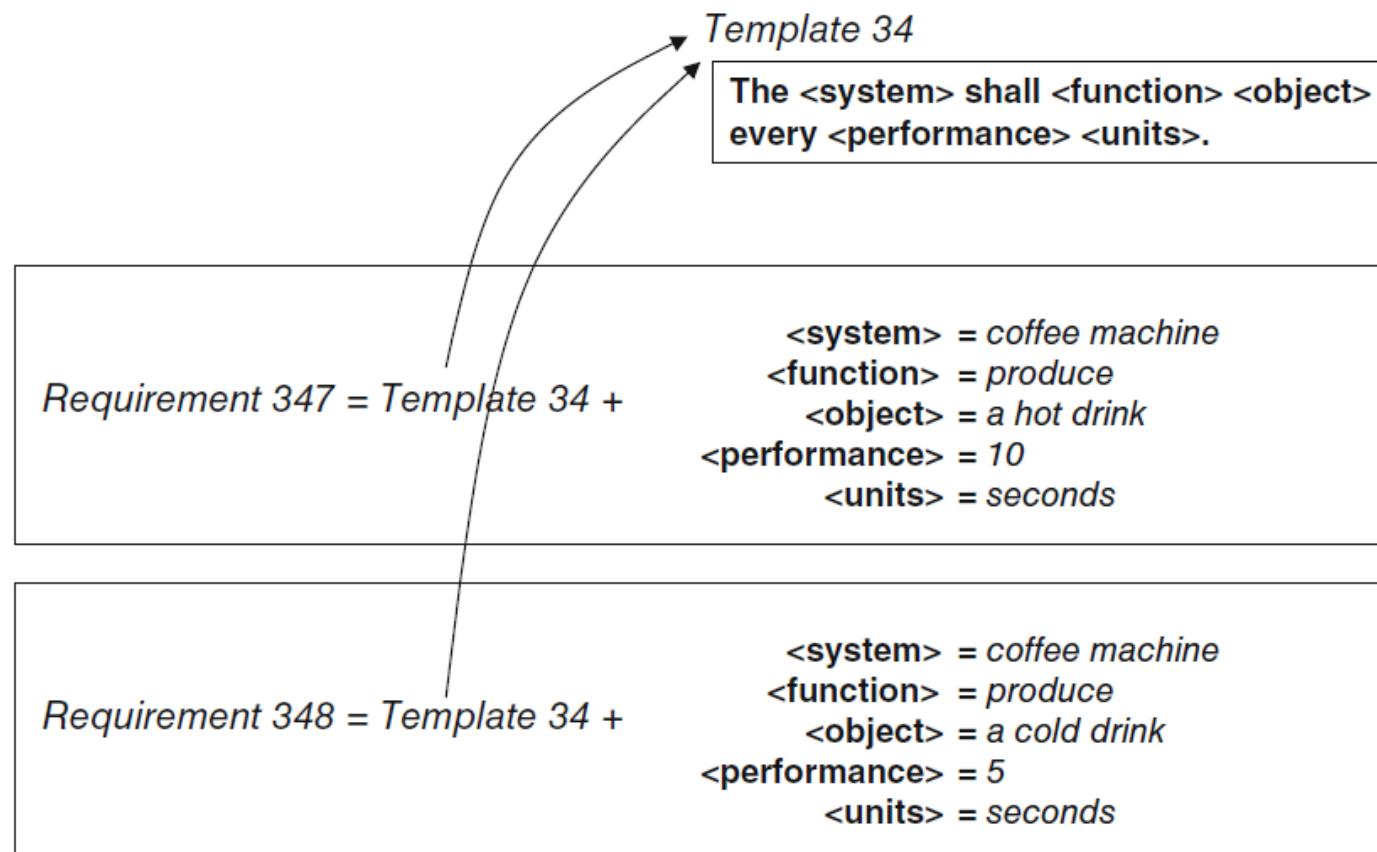
Extensions Attributes in SysML (as of SysML 1.6)

Table E-4: Requirement property enumeration types

| Enumeration | Enumeration Literals | Example Description |
|------------------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RiskKind | High | High indicates an unacceptable level of risk |
| | Medium | Medium indicates an acceptable level of risk |
| | Low | Low indicates a minimal level of risk or no risk |
| VerificationMethodKind | Analysis | Analysis indicates that verification will be performed by technical evaluation using mathematical representations, charts, graphs, circuit diagrams, data reduction, or representative data. Analysis also includes the verification of requirements under conditions, which are simulated or modeled; where the results are derived from the analysis of the results produced by the model. |
| | Demonstration | Demonstration indicates that verification will be performed by operation, movement or adjustment of the item under specific conditions to perform the design functions without recording of quantitative data. Demonstration is typically considered the least restrictive of the verification types. |
| | Inspection | Inspection indicates that verification will be performed by examination of the item, reviewing descriptive documentation, and comparing the appropriate characteristics with a predetermined standard to determine conformance to requirements without the use of special laboratory equipment or procedures. |
| | Test | Test indicates that verification will be performed through systematic exercising of the applicable item under appropriate conditions with instrumentation to measure required parameters and the collection, analysis, and evaluation of quantitative data to show that measured parameters equal or exceed specified requirements. |

Custom Templates or Boilerplates

- Collect patterns from the organization and type of product to create a collection of requirements

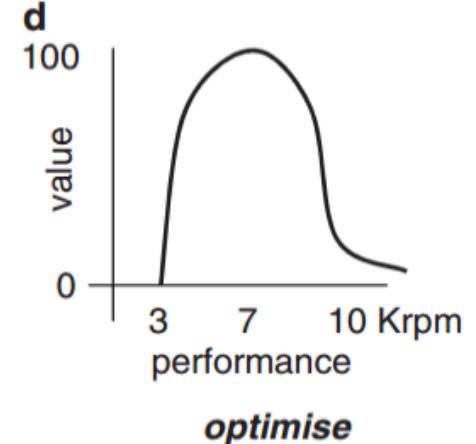
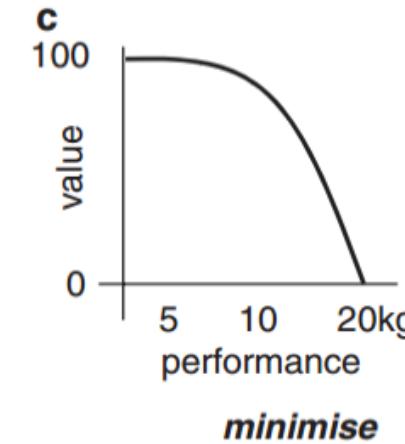
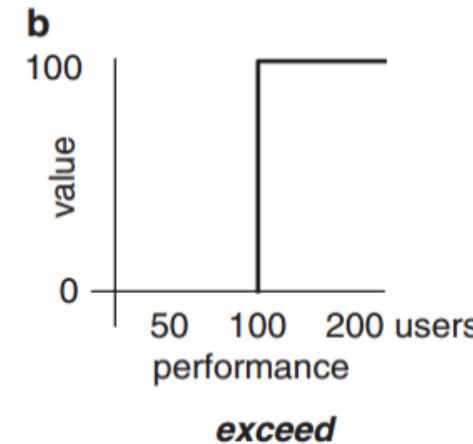
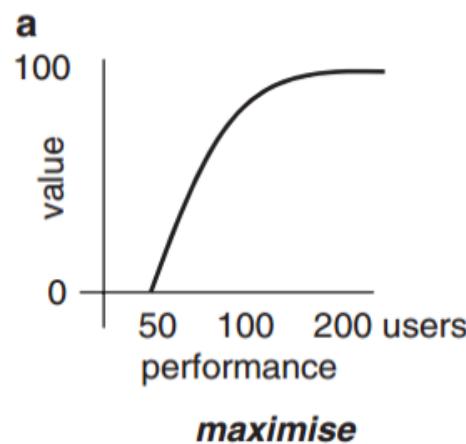


Examples (Hull Ch 4)

| | |
|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| Type of constraint | Boiler-plate |
| Performance/ capability | The <system> shall be able to <function> <object> not less than <performance> times per <units> . |
| Performance/ capability | The <system> shall be able to <function> <object> of type <qualification> within <performance> <units> . |
| Performance/ capacity | The <system> shall be able to <function> not less than <quantity> <object> . |
| Performance/ timeliness | The <system> shall be able to <function> <object> within <performance> <units> from <event> . |
| Performance/ periodicity | The <system> shall be able to <function> not less than <quantity> <object> within <performance> <units> . |
| Interoperability/ capacity | The <system> shall be able to <function> <object> composed of not less than <performance> <units> with <external entity> . |
| Sustainability/ periodicity | The <system> shall be able to <function> <object> for <performance> <units> every <performance> <units> . |
| Environmental/ operability | The <system> shall be able to <function> <object> while <operational condition> . |

Requirements Value

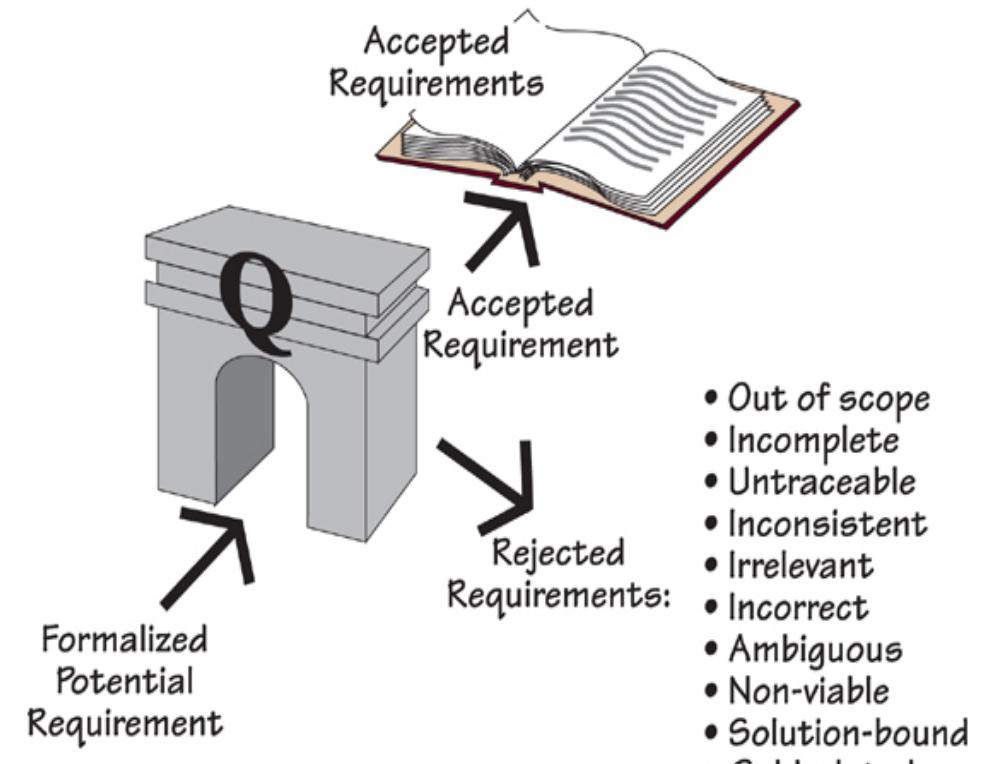
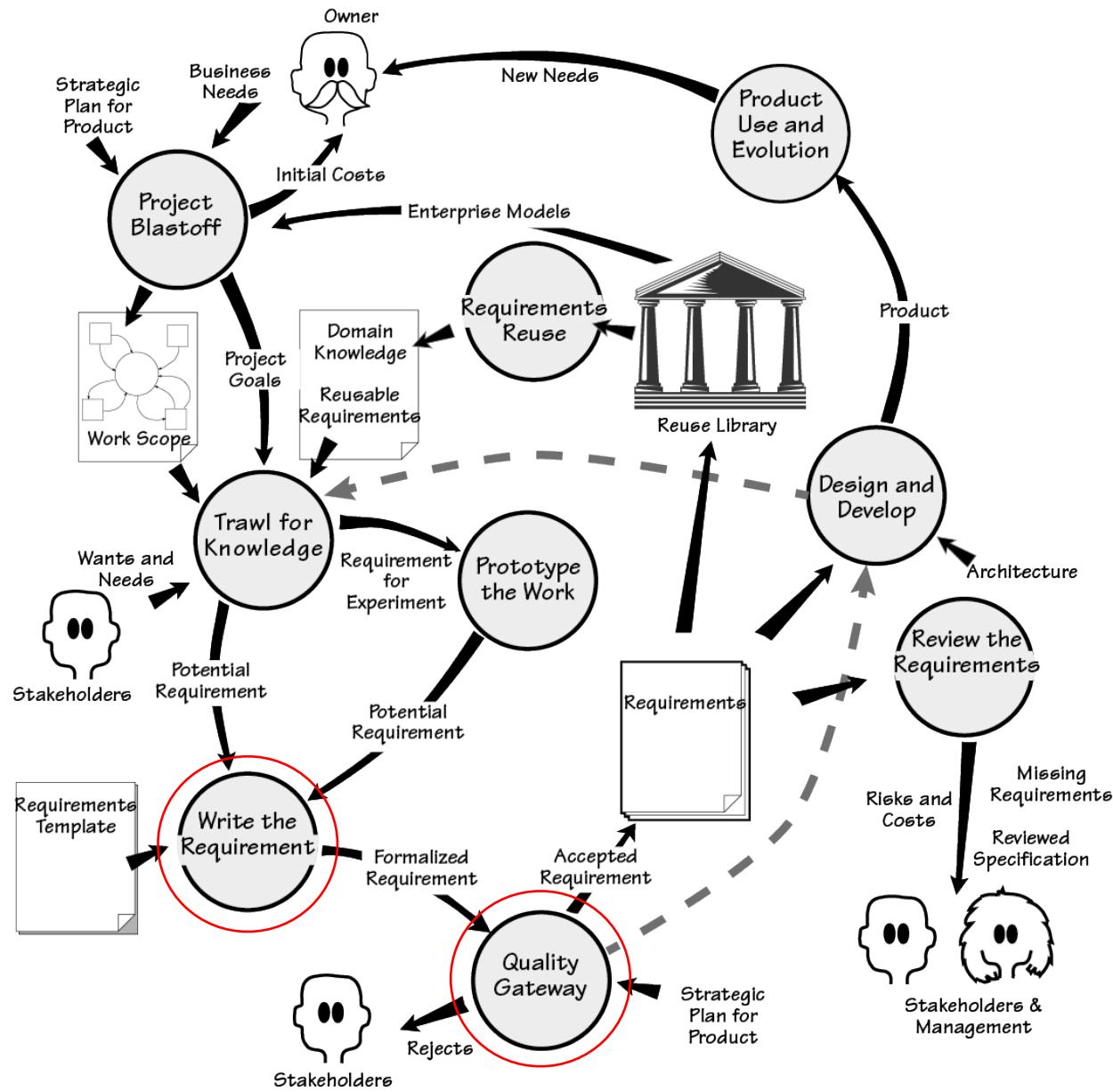
- Functional requirements are non-negotiable. If they are not met , the product is of no use
- Other requirements may be negotiable
 - One approach is to provide performance values (Mandatory, Desired, Best)
 - Another proposed approach is to represent the value of a requirement value by supplying a function that maps performance to value



Summary

- Make requirements readable by applying existing rules
- Check INCOSE and IEEE for additional recommendations and up-to-date information
- Requirements are not only for humans, NLP engines can consume requirements too!!
- When gathering requirements paraphrase and see if it make sense
- Obtain agreement and consensus from stakeholders
- Try to create patterns for requirements

Where are we?



Volere Requirement Card

| The type from the template | List of events / use cases that need this requirement |
|-----------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------|
| Requirement #: Unique id | Requirement Type: |
| Event/use case #: | |
| Description: A one sentence statement of the intention of the requirement | |
| Rationale: A justification of the requirement | |
| Source: Who raised this requirement? | |
| Fit Criterion: A measurement of the requirement such that it is possible to test if the solution matches the original requirement | |
| Customer Satisfaction: | Customer Dissatisfaction: |
| Dependencies: A list of other requirements that have some dependency on this one | Conflicts: Other requirements that cannot be implemented if this one is |
| Supporting Materials: — Pointer to documents | |
| History: Creation, changes, that illustrate and explain this requirement | |

Volere
Copyright © Atlantic Systems Guild

Degree of stakeholder happiness if this requirement is successfully implemented.

Scale from 1 = uninterested to 5 = extremely pleased.

Measure of stakeholder unhappiness if this requirement is not part of the final product.
Scale from 1 = hardly matters to 5 = extremely displeased.

Example of a Finished Requirement

Requirement #: **75**

Requirement Type: **9**

Event/BUC/PUC #: **7, 9**

Description: **The product shall record all the roads that have been treated**

Rationale: **To be able to schedule untreated roads and highlight potential**

Originator: **Arnold Snow - Chief Engineer**

Fit Criterion: **The recorded treated roads shall agree with the drivers' road treatment logs and shall be up to date within 30 minutes of the completion of the road's treatment**

Customer Satisfaction: **3**

Customer Dissatisfaction: **5**

Dependencies: **All requirements using road and scheduling data**

Conflicts: **105**

Supporting Materials:

Work context diagram, terms definitions in section 5

History: **Created February 29, 2010**

Volere

Copyright © Atlantic Systems Guild

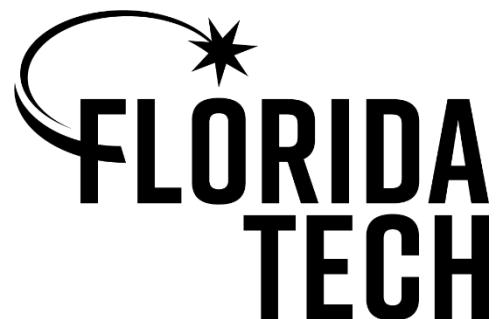
SYS 5460: Functional (cont.) & Non-Functional Requirements

Contents

- Overview of the Engineering Design Process
- Functional and Non-functional Requirement Formal Definitions
- Examples
 - Reliability
 - Response rate

Department of Computer & Engineering Sciences

College of Engineering
Florida Institute of Technology



Using Requirements Templates or Boiler Plates

- [when?] [under what conditions?] THE SYSTEM SHALL | SHOULD | WILL <process> <thing to be processed>[<process detail>*]

[Condition] [Subject] [Action] [Object] [Constraint]

EXAMPLE: When signal x is received [Condition], the system [Subject] shall set [Action] the signal x received bit [Object] within 2 seconds [Constraint].

Or

[Condition] [Action or Constraint] [Value]

EXAMPLE: At sea state 1 [Condition], the Radar System shall detect targets at ranges out to [Action or Constraint] 100 nautical miles [Value].

Or

[Subject] [Action] [Value]

EXAMPLE: The Invoice System [Subject], shall display pending customer invoices [Action] in ascending order [Value] in which invoices are to be paid.

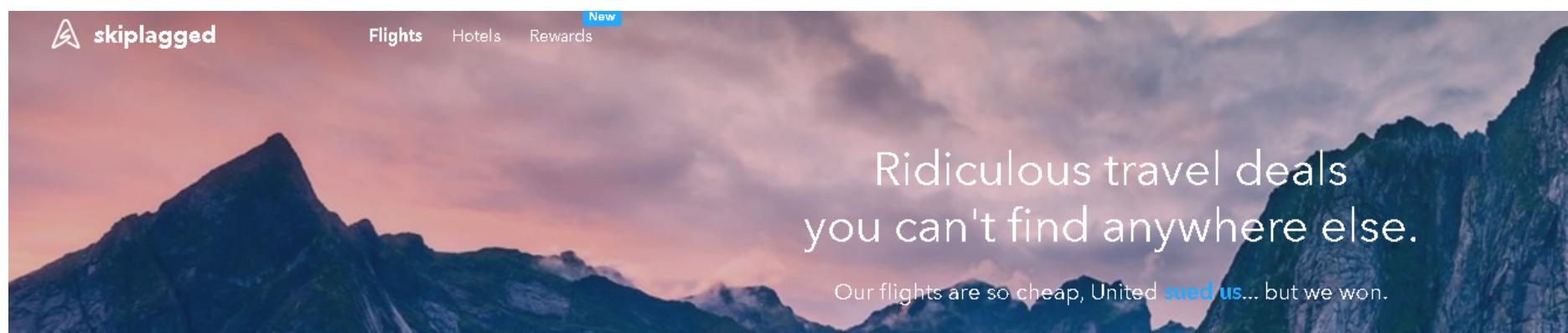
■ For Stakeholder capabilities

The <stakeholder type> shall be able to <capability> within <performance> of <event> while <operational condition>.

The <stakeholder> shall not be placed in breach of <applicable law>.

The weapons operator shall be able to fire a missile within 3 seconds of radar sighting while in severe sea conditions.

E.g. The ambulance driver shall not be placed in breach of national road regulations.



Find flights the airlines don't want you to see.
We're exposing **loopholes** in airfare pricing to save you money.

■ For System capabilities

The **<system>** shall **<function>**
not less than **<quantity>** **<object>**
while **<operational condition>**.

*E.g. The communications system shall sustain telephone contact
with not less than 10 callers
while in the absence of external power.*

The **<system>** shall **<function>** **<object>**
every **<performance>** **<units>**.

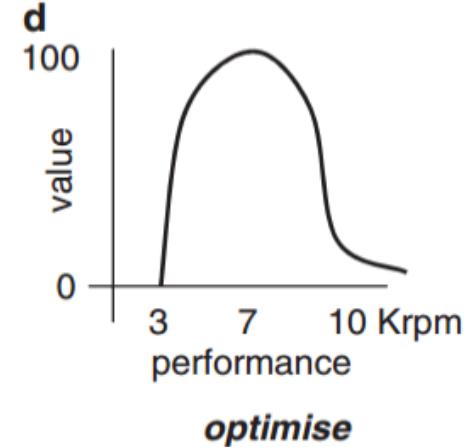
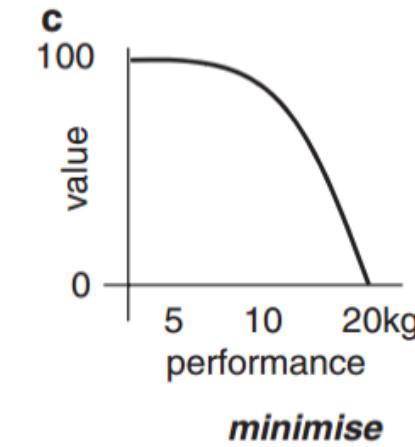
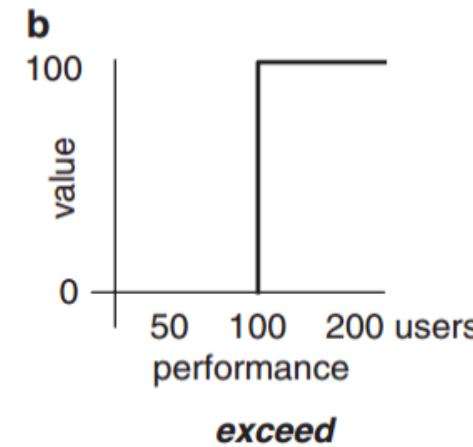
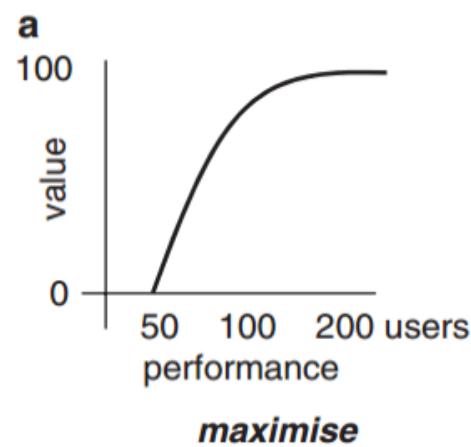
*E.g. The coffee machine shall produce a hot drink
every 10 seconds.*

Examples (Hull Ch 4)

| | |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| Type of constraint | Boiler-plate |
| Performance/ capability | The <system> shall be able to <function> <object> not less than <performance> times per <units>. |
| Performance/ capability | The <system> shall be able to <function> <object> of type <qualification> within <performance> <units>. |
| Performance/ capacity | The <system> shall be able to <function> not less than <quantity> <object>. |
| Performance/ timeliness | The <system> shall be able to <function> <object> within <performance> <units> from <event>. |
| Performance/ periodicity | The <system> shall be able to <function> not less than <quantity> <object> within <performance> <units>. |
| Interoperability/ capacity | The <system> shall be able to <function> <object> composed of not less than <performance> <units> with <external entity>. |
| Sustainability/ periodicity | The <system> shall be able to <function> <object> for <performance> <units> every <performance> <units>. |
| Environmental/ operability | The <system> shall be able to <function> <object> while <operational condition>. |

Requirements Value

- Functional requirements are non-negotiable. If they are not met , the product is of no use
- Other requirements may be negotiable
 - One approach is to provide performance values (Mandatory, Desired, Best)
 - Another proposed approach is to represent the value of a requirement value by supplying a function that maps performance to value



Technical Processes in the Design Stage

| Technical process | Purpose |
|----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. Stakeholder requirements definition | “Define the requirements for a system that can provide the services needed by users and other stakeholders in a defined environment” IEEE and ISO/IEC (2008, p. 36) |
| 2. Requirements analysis | “Transform the stakeholder, requirement-driven view of desired services into a technical view of a required product that could deliver those services” IEEE and ISO/IEC (2008, p. 39) |
| 3. Architectural design | “Synthesize a solution that satisfies system requirements” IEEE and ISO/IEC (2008, p. 40) |
| 4. Implementation | “Realize a specified system element” IEEE and ISO/IEC (2008, p. 43) |
| 5. Integration | “Assemble a system that is consistent with the architectural design” IEEE and ISO/IEC (2008, p. 44) |
| 6. Verification | “Confirm that the specified design requirements are fulfilled by the system” IEEE and ISO/IEC (2008, p. 45) |
| 7. Transition | “Establish a capability to provide services specified by stakeholder requirements in the operational environment” IEEE and ISO/IEC (2008, p. 46) |
| 8. Validation | “Provide objective evidence that the services provided by a system when in use comply with stakeholders’ requirements, achieving its intended use in its intended operational environment” IEEE and ISO/IEC (2008, p. 47) |

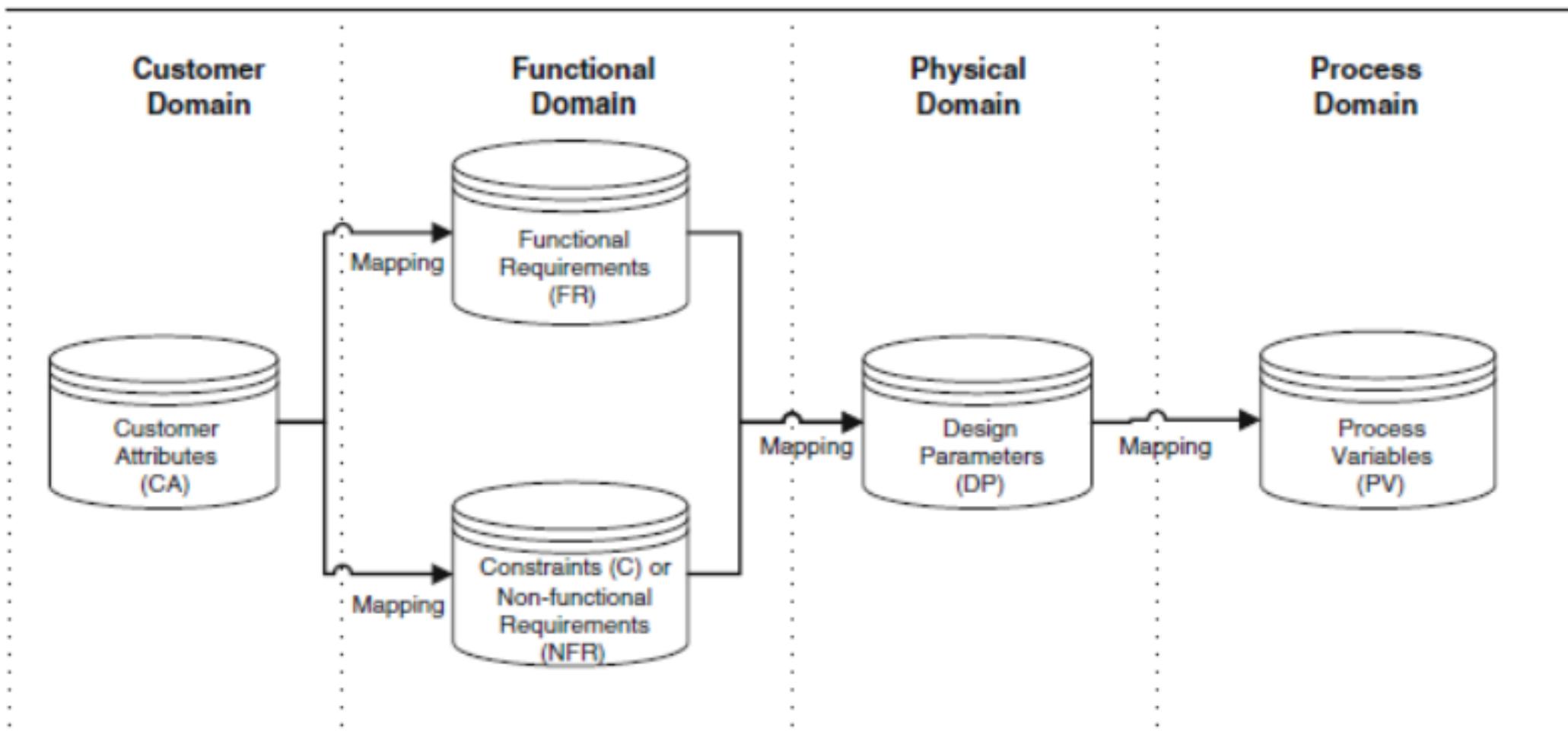
Functional Requirement

- A statement that identifies what a product or process must accomplish to produce required behavior and/or results.
- A requirement that specifies a function that a system or system component must be able to perform. (IEEE and ISO/IEC 2010, p. 153)
- **Functional** requirements have the following essential characteristics:
 - Define what the system should do
 - Be action oriented
 - Describe tasks or activities
 - Associated with the transformation of inputs to outputs

Non-Functional Requirements

- A software requirement that describes not what the software will do but how the software will do it (IEEE and ISO/IEC 2010, p. 231)
- Properties or qualities the product must have to facilitate its functionality (Robertson and Robertson 2005, p. 146)
- Non-Functional requirements have the following essential characteristics:
 - Define a property or quality that the system should have
 - Can be subjective, relative, and interacting
 - Describe how well the systems must operate
 - Are associated with the entire system

Formal Process of Engineering Design (DSM)



Stakeholder Requirements

Requirements Analysis

System Architecture

Production of the Solution

Axiomatic Design Methodology

- Independence Axiom: each functional requirement should be satisfied **without affecting any other functional requirement**.
- Information Axiom: The system design that requires the **least amount of information** to fulfill the design parameters is preferred.

$$a_{ij} = 1 \text{ if } FR\ i \text{ is satisfied by } DP\ j$$

Equation for functional requirements

$$[FR] = [A][DP]$$



FR: Functional requirement

A: Design matrix

DP: Design parameters

$$[A] = \begin{vmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{vmatrix}$$

$$FR_1 = A_{11}DP_1 + A_{12}DP_2 + A_{13}DP_3$$

$$FR_2 = A_{21}DP_1 + A_{22}DP_2 + A_{23}DP_3$$

$$FR_3 = A_{31}DP_1 + A_{32}DP_2 + A_{33}DP_3$$

What type of matrix is A to satisfy axiom 1

Examples

Canon TS9120 Wireless All-In-One Printer with Scanner and Copier: Mobile and Tablet Printing, with Airprint(TM) and Google Cloud Print compatible, Gray

by Canon

★★★★★ 530 customer reviews | 445 answered questions

List Price: \$199.99

Price: \$59.99 prime

You Save: \$140.00 (70%)

Get \$70 off instantly: Pay \$0.00 upon approval for the Amazon Prime Rewards Visa Card.

Color: Gray



Style: Printer

Printer **Printer and Ink Bundle** **Printer, Ink and Paper Bundle**

- Inspire your creativity with prints that will impress. From stunning photographs to detailed documents, put the 6-color individual ink system to work and never compromise on speed or quality.
- Enjoy the simplicity of connecting your smartphone, tablet and all your favorite devices with ease. Print hassle free - whether from the cloud, through Bluetooth, from social media or even on the go.
- The PIXMA TS9120 Wireless printer is the all-in-one that fits perfectly anywhere in the home, and looks great with any decor thanks to its two-tone design and multiple color options.
- With intuitive features like the 5.0" LCD touchscreen and enhanced user interface, Bluetooth printing, & document removal reminder, it's clear to see why the PIXMA TS9120 makes printing a breeze.



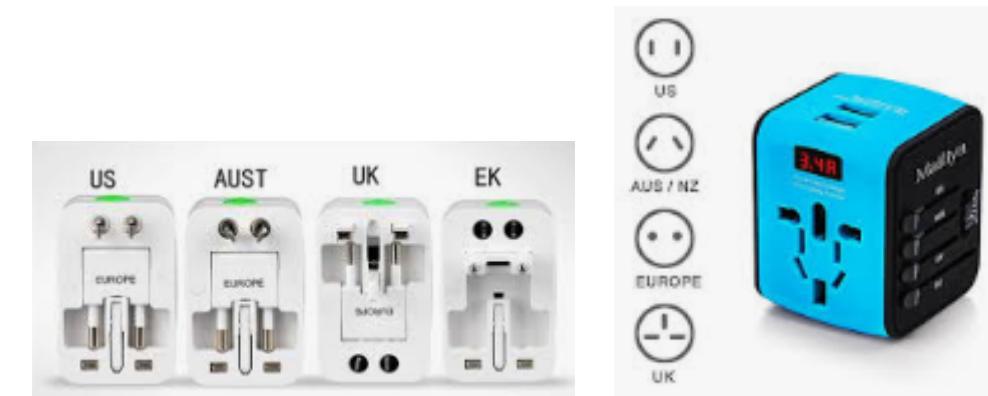
Re-Caulk, the Easy Way

Allway's new 3-in-1 Caulk Tool helps remove and re-apply Caulk Quickly and neatly. Stainless Steel Caulk Remover pulls out the old Caulk bead with either a push or pull motion.

ANYONE CAN DO IT. IT'S EASY.

Triangular rubber blade is adjustable. There is a choice of 3 rounded tips to provide the desired profile to smooth the freshly applied Caulk bead, neatly around the tub, shower, sink, window, or exterior trim.

Allway Tools
Better by Design

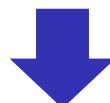


Independence and Design Complexity

- The independence axiom may be used to evaluate design complexity
- Systems exhibit complexity as a result of excessive interaction between components within the system design
- If the number of DPs is less than the number of FRs:
 - The design has added complexity because DPs are satisfying more than one FR
 - The FR has not been satisfied



Your system requirement tool shall assist you with this

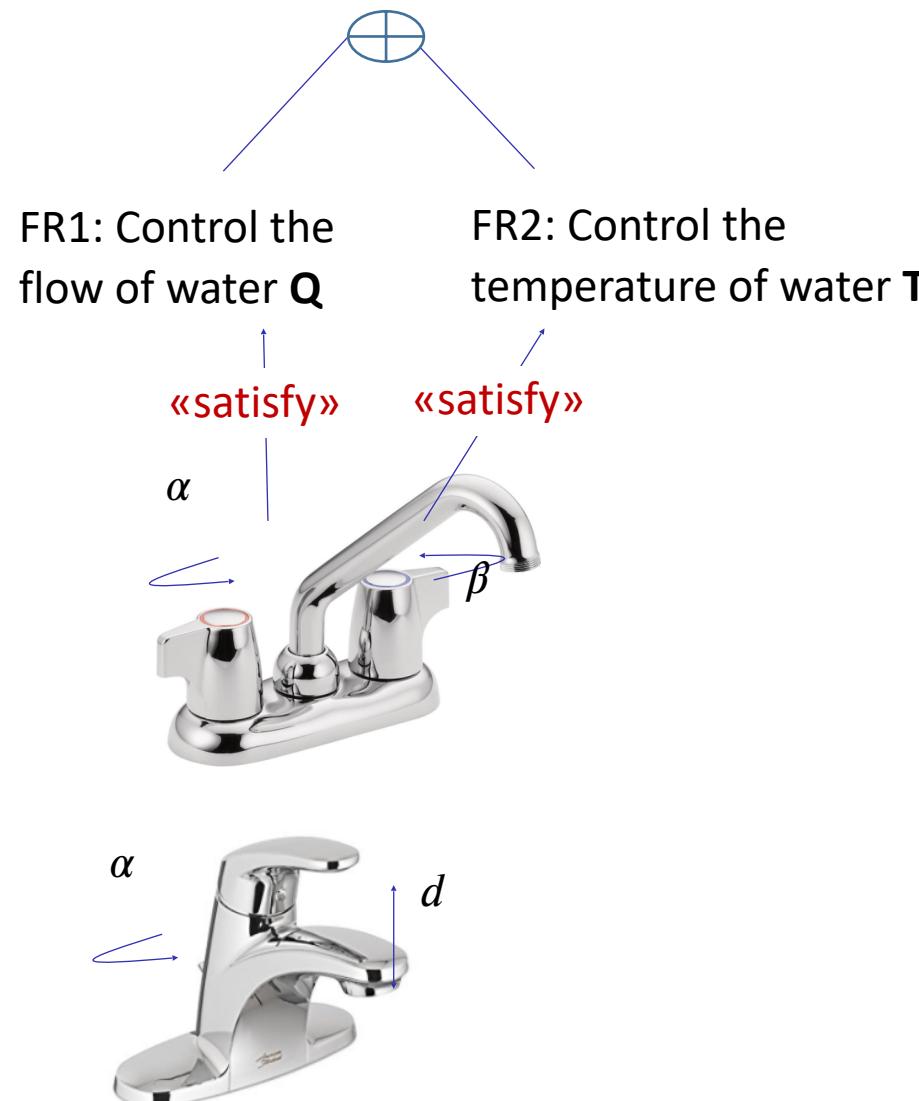


Relationships are build in the system using a requirements tool or MBSE tool

| Relationship Name | Keyword | Source-> Target Interpretation |
|--------------------|------------------|------------------------------------------------------------------------------|
| Satisfy | «satisfy» | «model element» satisfy «requirement» |
| Verify | «verify» | «test case» verify «requirement» |
| Refine | «refine» | «model element» refine «requirement» «requirement» refine «model element» |
| Derive Requirement | «deriveRqt» | «requirement» derived from «requirement» |
| Copy | «copy» | «requirement» copied from «requirement» |
| Trace | «Trace» | «requirement» traced from «requirement» |
| Containment | (Crosshair icon) | «requirement» is contained in «requirement» |

Example: Water Faucet

The user shall be able to control the temperature and flow of water



Requirements and System Architecting

- Requirements should not anticipate the solution
- Requirements describe the what
- Architecture describes the how
- Solution-free requirements (arguably) may not be feasible in practice
- Every time we satisfy a requirement with an element we are deriving logical architectural components
- Each new component leads to a new requirements that are related to aspects of the solution
- Example:
 - The system shall transport 2 people from point A to point B
 - Solution could be a ground transport (say car->constraint)
 - Then we need to determine requirements specific to car
- The Architecture that lie in the background of the solution is the base system architecture

Non-Functional Requirements (Robertson Ch11)

- Non-functional requirements are properties that the **functionality** must have
- Non-functional requirements do not alter the product's essential functionality
 - Look and Feel: the spirit of the product's appearance
 - Usability and Humanity: the product's ease of use,
 - Performance: how fast, how safe, how many, how available, and how accurate the functionality must be
 - Operational: the operating environment of the product,
 - Maintainability and Support: expected changes, and the time needed to make them;
 - Security: access, confidentiality, recoverability, and auditability of the product
 - Cultural and Political: special requirements that come about because of the culture and customs of people who can come in contact with the product
 - Legal: the laws and standards that apply to the product

Look and Feel (Req Type 10)

- Appear to be simple to use
- Approachable, so people do not hesitate to use it
- Authoritative, user feel that they can rely on it and trust it
- Conforming to the client's other products
- Attractive to children
- Unobtrusive , so that people are not aware of it
- Innovative and appearing to be state of the art
- Professional looking
- Exciting
- Cool

Stakeholder Requirement:

The product shall be comply with windows user experience interactions guidelines for windows 8

Performance (Req Type 12)

- Speed to complete a task
- Accuracy of the results
- Safety of the operator
- Volumes of data to be stored
- Ranges of allowable values
- Throughput , rate of transactions
- Efficiency of resource usage
- Reliability (MTBF)
- Availability (uptime)
- Fault tolerance
- Scalability

Stakeholder Requirement:

The product shall schedule de-icing activities , so that the minimum necessary amounts of de-icing material are spread on roads

Operational and Environmental (Req Type 13)

- Operating environment (water, land, air ,space)
- Operating conditions (temp, pressure, wind speeds, tide severity, illumination)
- Partner of collaborating systems
- Threatening systems

Stakeholder Requirement:

The product shall survive being dropped from shoulder height

Maintainability and Support (Req Type 14)

- Expected changes in the product in the following areas:

- Organization
- Environment
- Laws that apply to the product
- Business rules

Stakeholder Requirement:

The product shall be translated into various foreign languages. As of yet the languages are unknown

Security (Req Type 15)

- Access: Data and functionality accessible to authorized users. Deny access to unauthorized people
- Privacy: Data is protected from unauthorized or accidental disclosure
- Integrity: Product data is the same as the source, or authority of the data. Protected from corruption

Stakeholder Requirement:

The product shall ensure that only authorized users have access to the products data

The product shall ensure its road temperature data corresponds to the data transmitted by the weather station

Cultural (Req Type 16)

- Avoid use of symbols associated with religion, politics , etc.
- Incorporate widely acceptable phrases when using foreign languages (e.g. Spanish, French, English)

Stakeholder Requirement:

The product shall not use any terms or icons that might possible offend anyone on the planet

Legal (Req Type 17)

- Examine adjacent systems or actors
- Consider legal rights or applicable laws:
 - Privacy protection
 - Non-disclosure
 - Consumer protections
 - Warranties

Stakeholder Requirement:

The product shall comply with the Americans with Disabilities Act of 1990 as amended

Non-Functional Requirement and System “ilities”

- There are over 200 recognized system *ilities* in systems engineering
- List compiled by Adams(2015)

| Accessibility | Degradation of service | Modularity | Security |
|--------------------------|------------------------|-----------------------------|--------------------------|
| Accountability | Dependability | Naturalness | Sensitivity |
| Accuracy | Development cost | Nomadicity | Similarity |
| Adaptability | Development time | Observability | Simplicity |
| Additivity | Distributivity | Off-peak period performance | Software cost |
| Adjustability | Diversity | Operability | Software production time |
| Affordability | Domain analysis cost | Operating cost | Space boundedness |
| Agility | Domain analysis time | Peak-period performance | Space performance |
| Auditability | Efficiency | Performability | Specificity |
| Availability | Elasticity | Performance | Stability |
| Buffer space performance | Enhanceability | Planning cost | Standardizability |
| Capability | Evolvability | Planning time | Subjectivity |
| Capacity | Execution cost | Plasticity | Supportability |
| Clarity | Extensibility | Portability | Surety |
| Code-space performance | External consistency | Precision | Survivability |
| Cohesiveness | Fault-tolerance | Predictability | Susceptibility |
| Commonality | Feasibility | Process management time | Sustainability |

68

Sample of Non-Functional Requirements (1)

| | | | |
|----------------------------|----------------------|-----------------------|---------------------|
| Communication cost | Flexibility | Productivity | Testability |
| Communication time | Formality | Project stability | Testing time |
| Compatibility | Generality | Project tracking cost | Throughput |
| Completeness | Guidance | Promptness | Time performance |
| Component integration cost | Hardware cost | Prototyping cost | Timeliness |
| Component integration time | Impact analyzability | Prototyping time | Tolerance |
| Composability | Independence | Reconfigurability | Traceability |
| Comprehensibility | Informativeness | Recoverability | Trainability |
| Conceptuality | Inspection cost | Recovery | Transferability |
| Conciseness | Inspection time | Reengineering cost | Transparency |
| Confidentiality | Integrity | Reliability | Understandability |
| Configurability | Inter-operability | Repeatability | Uniform performance |
| Consistency | Internal consistency | Replaceability | Uniformity |
| Controllability | Intuitiveness | Replicability | Usability |
| Coordination cost | Learnability | Response time | User-friendliness |
| Coordination time | | Responsiveness | Validity |

64

Cumulative
total:132

Sample of Non-Functional Requirements (2)

| | | | |
|--------------------------|-------------------------|-------------------------------|---------------|
| | Main-memory performance | | |
| Correctness | Maintainability | Retirement cost | Variability |
| Cost | Maintenance cost | Reusability | Verifiability |
| Coupling | Maintenance time | Risk analysis cost | Versatility |
| Customer evaluation time | Maturity | Risk analysis time | Visibility |
| Customer loyalty | Mean performance | Robustness | Wrappability |
| Customizability | Measurability | Safety | |
| Data-space performance | Mobility | Scalability | |
| Decomposability | Modifiability | Secondary-storage performance | |

33

Cumulative
total:165

Sample of Non-Functional Requirements (3)

| | | |
|-------------------|-----------------|----------------------|
| Analyzability | Demonstrability | Manageability |
| Anonymity | Durability | Performance |
| Atomicity | Effectiveness | Privacy |
| Attractiveness | Expandability | Provability |
| Augmentability | Expressiveness | Quality of service |
| Certainty | Extendability | Readability |
| Changeability | Functionality | Self-descriptiveness |
| Communicativeness | Immunity | Structuredness |
| Complexity | Installability | Suitability |
| Comprehensiveness | Integratability | Tailorability |
| Conformance | Legibility | Trustability |
| Debuggability | Likeability | Viability |
| Defensibility | Localizability | |

| | | |
|------------------|--------------------|-----------------|
| Degradability | Heterogeneity | Reproducibility |
| Deployability | Homogeneity | Resilience |
| Determinability | Interchangeability | Securability |
| Disposability | Manufacturability | Serviceability |
| Distributability | Producability | Ubiquity |
| Expandability | Repairability | |
| Fidelity | Repeatability | |

58

Cumulative
total:223

Example: Temperature



Moderna Announces Longer Shelf Life for its COVID-19 Vaccine Candidate at Standard Refrigerator Temperatures

November 16, 2020

Vaccine candidate now expected to remain stable at standard refrigerator temperatures of 2° to 8°C (36° to 46°F) for 30 days, up from previous estimate of 7 days

Shipping and long-term storage conditions at standard freezer temperatures of -20°C (-4°F) for 6 months

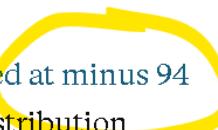
mRNA-1273 to be distributed using widely available vaccine delivery and storage infrastructure

No dilution required prior to vaccination

CAMBRIDGE, Mass.--(BUSINESS WIRE)--Nov. 16, 2020-- [Moderna, Inc.](#) (Nasdaq: MRNA), a biotechnology company pioneering messenger RNA (mRNA) therapeutics and vaccines to create a new generation of transformative medicines for patients, today announced new data showing that mRNA-1273, its COVID-19 vaccine candidate, remains stable at 2° to 8°C (36° to 46°F), the temperature of a standard home or medical refrigerator, for 30 days. Stability testing supports this extension from an earlier estimate of 7 days. mRNA-1273 remains stable at -20° C (-4°F) for up to six months, at refrigerated conditions for up to 30 days and at room temperature for up to 12 hours.

Pfizer has launched a pilot in Rhode Island, Texas, New Mexico, and Tennessee to test the distribution of its COVID-19 vaccine, which has to be kept at temperatures well below freezing.

Pfizer's vaccine candidate has to be shipped and stored at minus 94 degrees Fahrenheit (minus 70 degrees Celsius). Its distribution therefore relies on a "cold chain," a multipart pipeline that keeps the shots chilled, from manufacturer to injection.



PROJECT DELIVERABLES FOR NEXT WEEK

- Goals and project strategy
- Stakeholders
- Use Cases (Nominal or Off nominal scenarios)
- Requirements Tables:
 - Minimum of 15 requirements
 - Divided into functional and NF
- CONOPS (Describe the system)
- Verification and Validation
 - Verification Methods for 10 Requirements
 - Validation Methods for 5 Requirements
- 1 (poster file) Slide Marketing Material/ Display or Pitch
 - Font 8-10

References

- Chung, L., Nixon, B. A., Yu, E. S., & Mylopoulos, J. (2000). *Non-functional requirements in software engineering*. Boston: Kluwer Academic Publishers
- IEEE and ISO/IEC. (2008). *IEEE and ISO/IEC Standard 15288: Systems and software engineering —system life cycle processes*. New York: Institute of Electrical and Electronics Engineers and the International Organization for Standardization and the International Electrotechnical Commission.
- Adams, K. M. (2015). *Non-functional requirements in systems analysis and design* (Vol. 28). Heidelberg: Springer.

Introduction to AGILE & JIRA

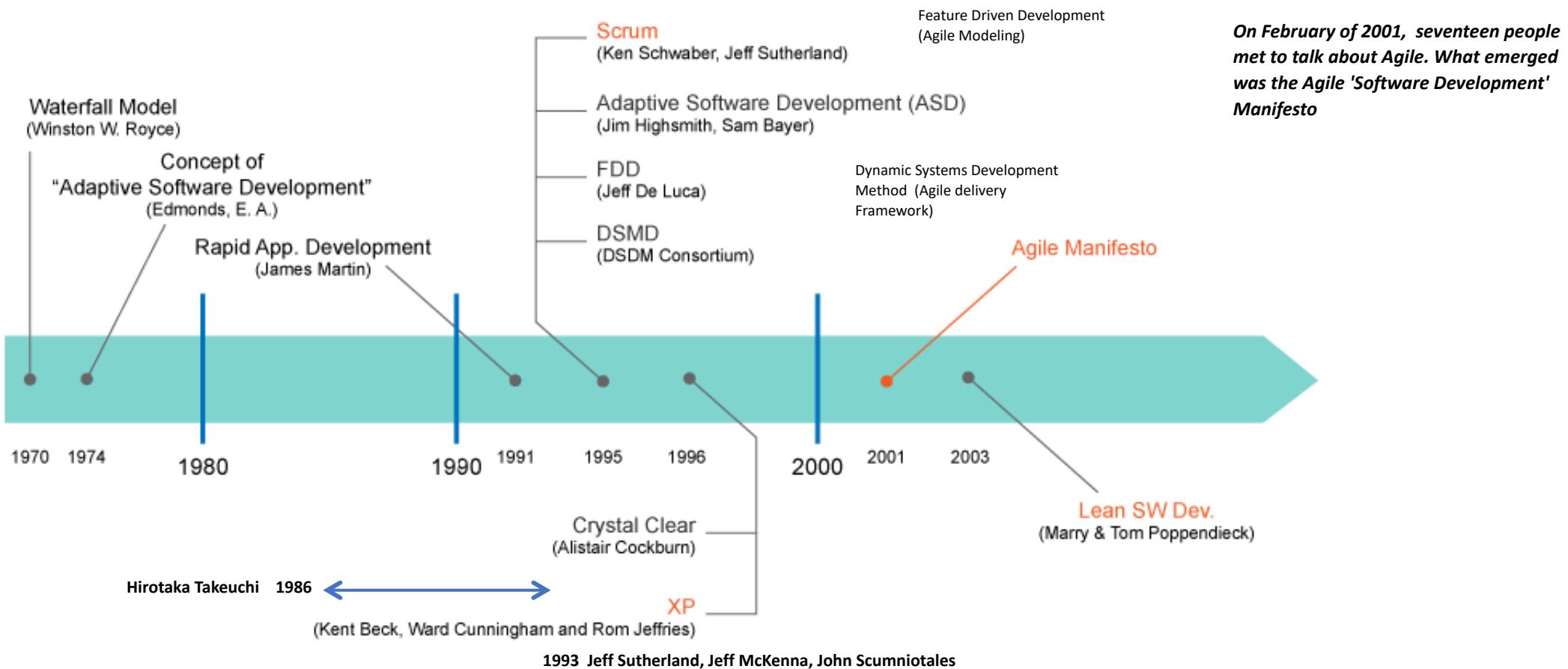
Juan Avendano, PhD



Agile History

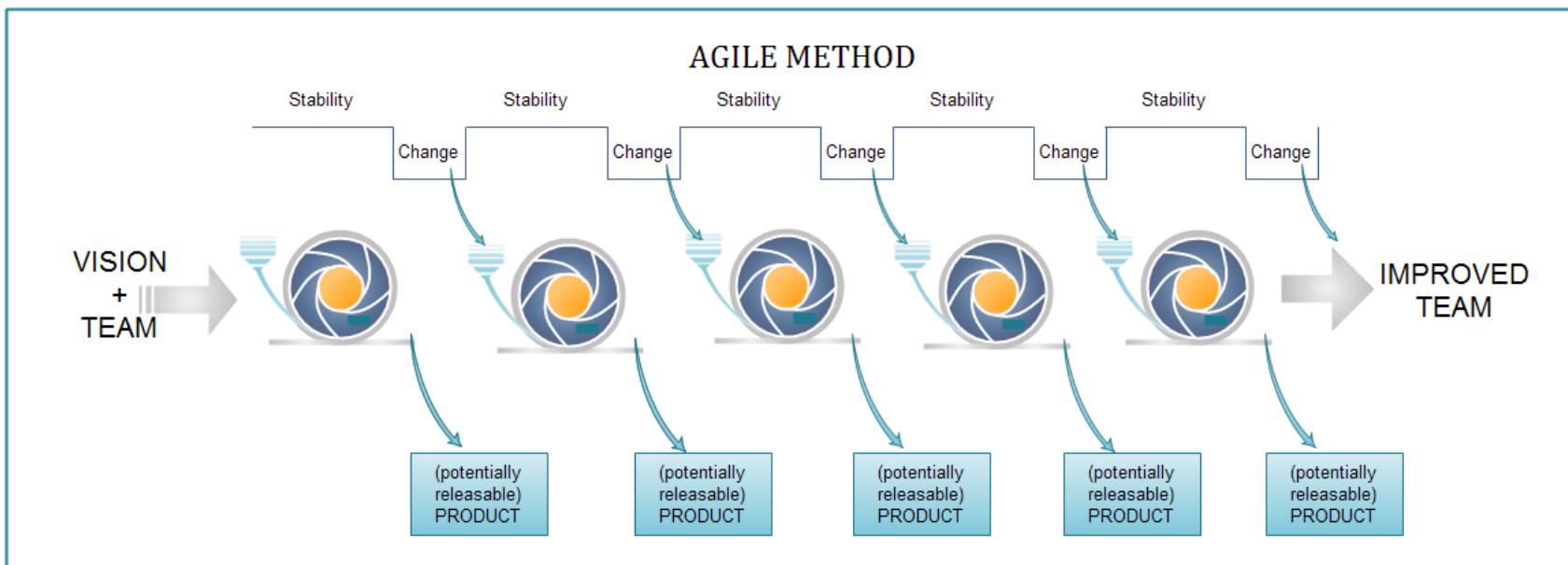
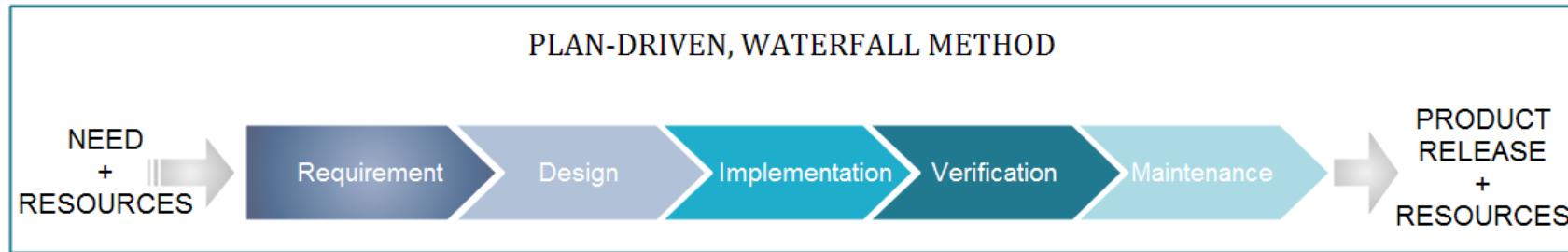
- ❑ In the early 1990s, software development faced a crisis. Industry experts estimated that the time between a validated business need and an actual application in production was about three years
- ❑ Within the space of three years, requirements, systems, and even entire businesses were likely to change. That meant that many projects ended up being cancelled partway through, and many of those that were completed didn't meet all the business's current needs, even if the project's original objectives were met
- ❑ In certain industries, the lag was far greater than three years. In aerospace and defense, it could be 20 or more years before a complex system went into actual use. The Space Shuttle program, which operationally launched in 1982, used information and processing technologies from the 1960s
- ❑ In 1990s, several technology leaders frustrated with these long lead times and decisions made early in a project that couldn't be changed late, began informal talks about ways to develop software more simply, without the process and documentation overhead of **Waterfall** and other popular software engineering techniques of the time

Agile History Timeline



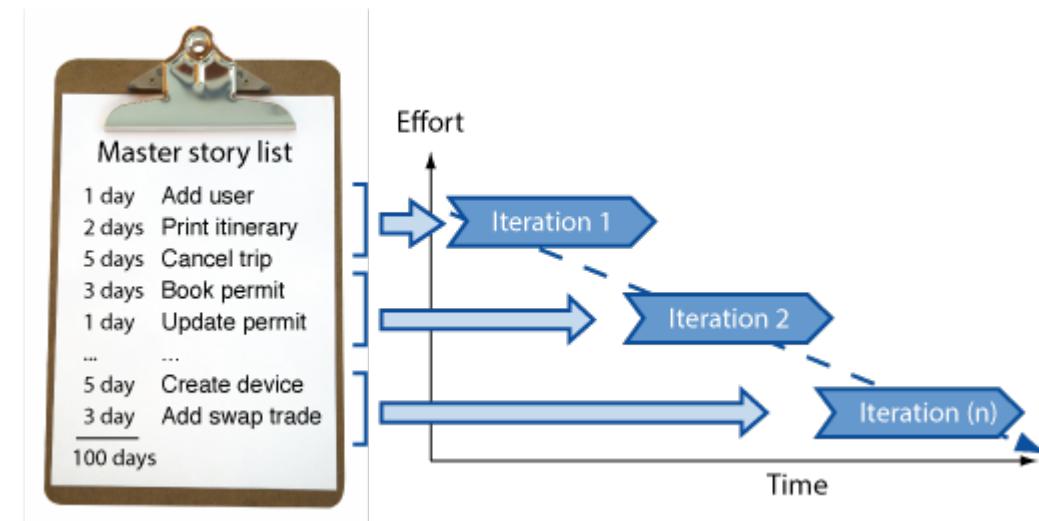
Evolution of Project Management

| Simple Comparison | |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Traditional Project Management | Modern Project Management (Agile) |
| Elements impacting execution of a project: <ul style="list-style-type: none">a) Planningb) Control | Elements impacting execution of a project: <ul style="list-style-type: none">a) Competitive environmentb) Creativity and Innovationc) Planningd) Control |
| Project Success is measured by having: <ul style="list-style-type: none">a) Well defined requirementsb) Approved budgetc) On-Time Delivery | Project Success is measured by: <ul style="list-style-type: none">a) Delivery of Business Value (what does customer want) <p>Less important are:</p> <ul style="list-style-type: none">a) Having well defined requirementsb) Budget constraintsc) Set Schedule <p>Today's world has a much higher level of uncertainty which makes it difficult to always start a project with well-defined requirements</p> |
| Requires a Project Manager | Project Manager Role is distributed among: <ul style="list-style-type: none">a) Product Ownerb) Scrum Masterc) Project Team |

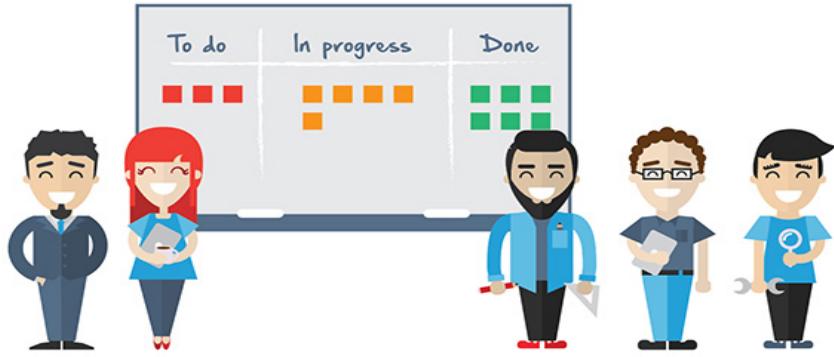


Understanding Agile

Agile is a time boxed, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end. It works by breaking projects down into little bits of user functionality called User Stories, prioritizing them, and then continuously delivering them in short two week cycles called Iteration



Scrum, Kanban & XP Agile Methodologies

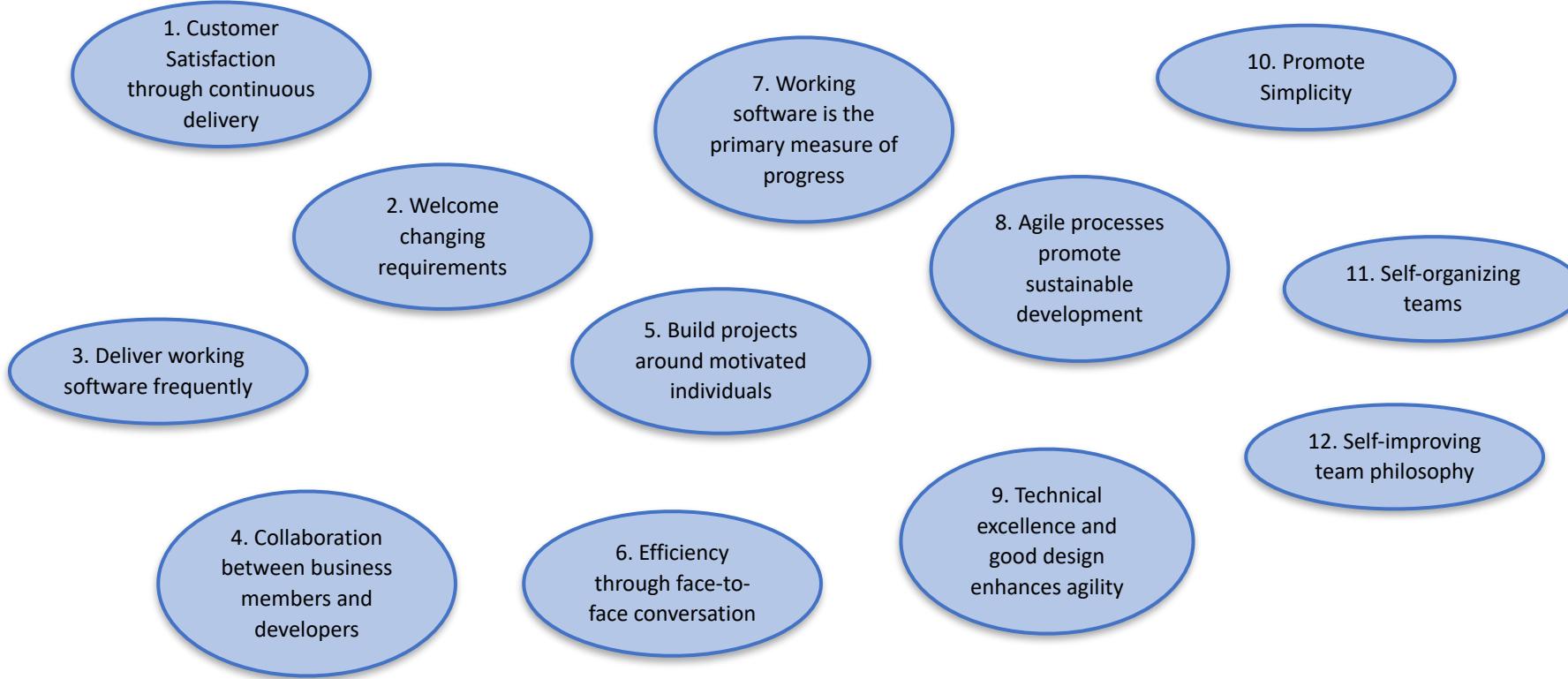


Kanban (an industrial engineer at Toyota, developed Kanban to improve manufacturing efficiency) is also a tool used to organize work for the sake of efficiency. Where Scrum limits the amount of time allowed to accomplish a particular amount of work (by means of sprints), Kanban limits the amount of work allowed in any one condition (only so many tasks can be ongoing, only so many can be on the to-do list.)

Scrum is a tool used to organize work into small, manageable pieces that can be completed by a cross-functional team within a prescribed time period (called a sprint, generally 2-4 weeks long) to plan, organize, administer, and optimize this process.

Extreme Programming (XP) is an agile software development framework that aims to produce higher quality software, while focusing on customer satisfaction by delivering what's needed when needed. Its guiding principles are: Communication, Simplicity, Feedback, Respect and Courage

Agile Manifesto



The Four Values of The Agile Manifesto

- Individuals and Interactions Over Processes and Tools
- Working Software Over Comprehensive Documentation
- Customer Collaboration Over Contract Negotiation
- Responding to change over following a Plan

XP and SCRUM





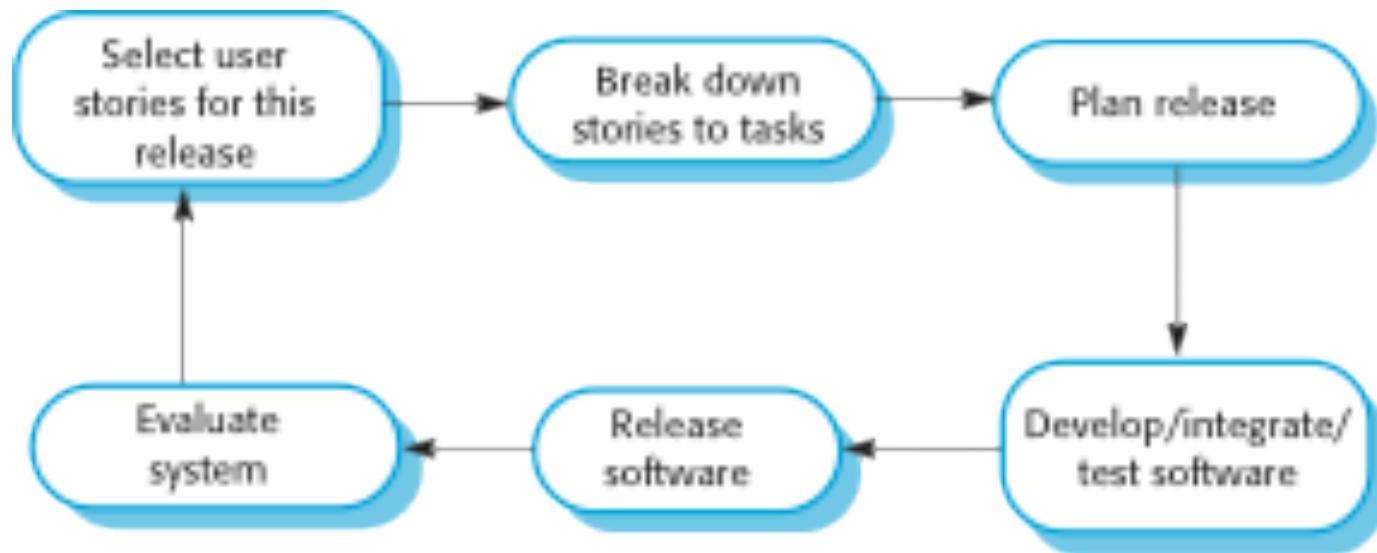
Difference between Scrum & XP

- Scrum teams typically work in iterations (called sprints) that are from two weeks to one month long. XP teams typically work in iterations that are one or two weeks long
- Scrum teams do not allow changes into their sprints. XP teams welcome changes as long as work has not started
- Extreme Programming teams work in a strict priority order. Scrum teams have flexibility to choose what prioritized features to work on
- Scrum doesn't prescribe any engineering practices. XP does (things like test-driven development, focus on automated testing, pair programming, simple design, refactoring, and so on)

Extreme programming

- ◊ A popular form of Agile
- ◊ Extreme Programming (XP) takes an ‘extreme’ approach to iterative development.
 - New versions may be built several times per day;
 - Increments are delivered to customers every 2 weeks;
 - All tests must be run for every build and the build is only accepted if tests run successfully.
- ◊ Customer involvement means full-time customer engagement with the team. - Specifications through user stories broken into tasks
- ◊ People not process : pair programming, collective ownership and a process that avoids long working hours.
- ◊ Regular system releases. - release set of user stories
- ◊ Maintaining simplicity through constant refactoring of code.

The extreme programming release cycle

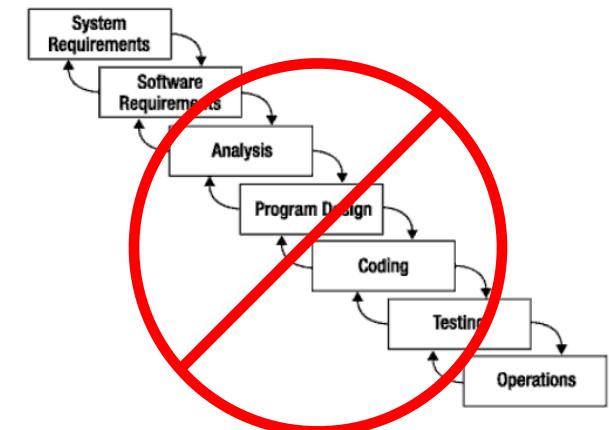


What is Scrum?

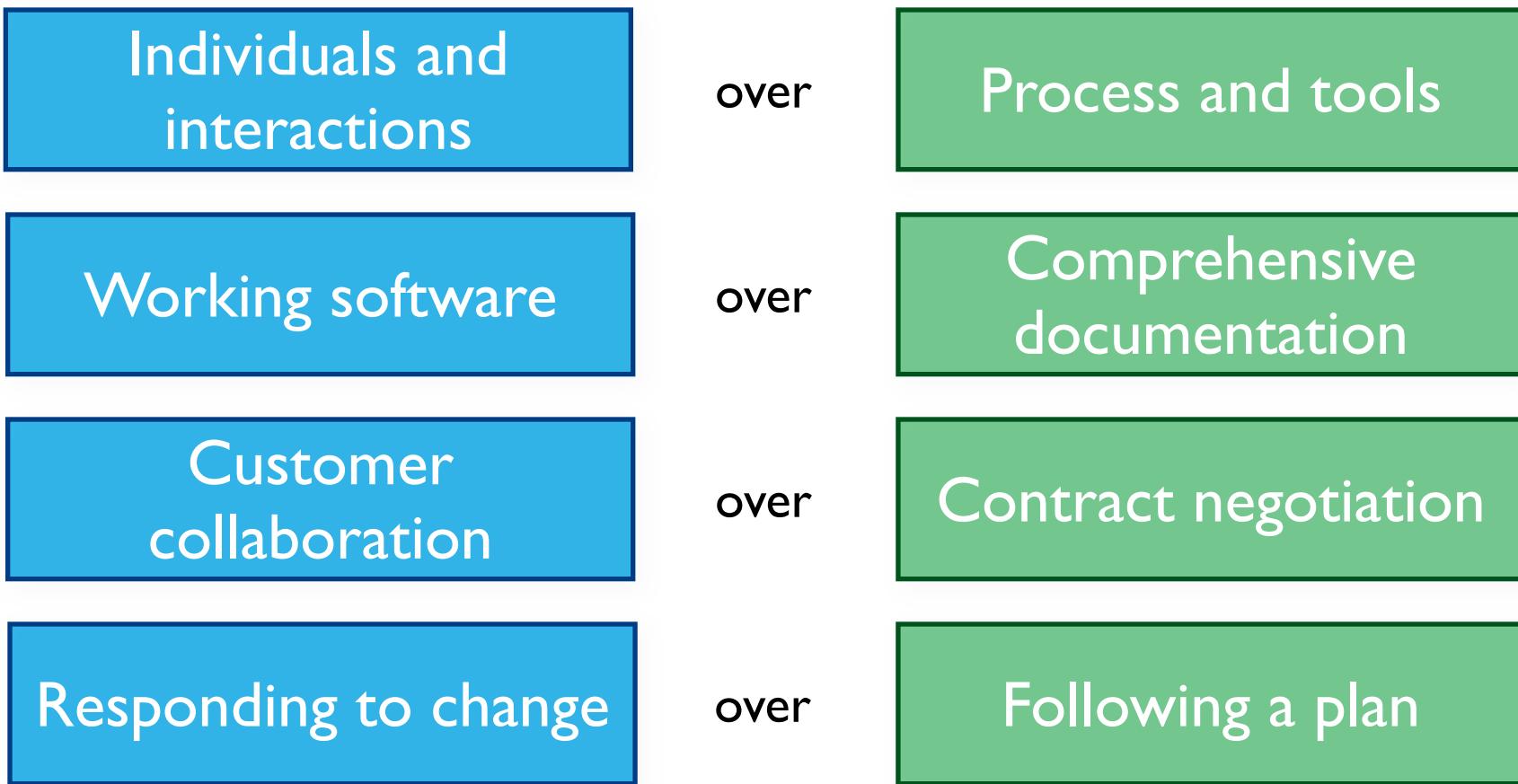
It's about common sense

- **Scrum:**

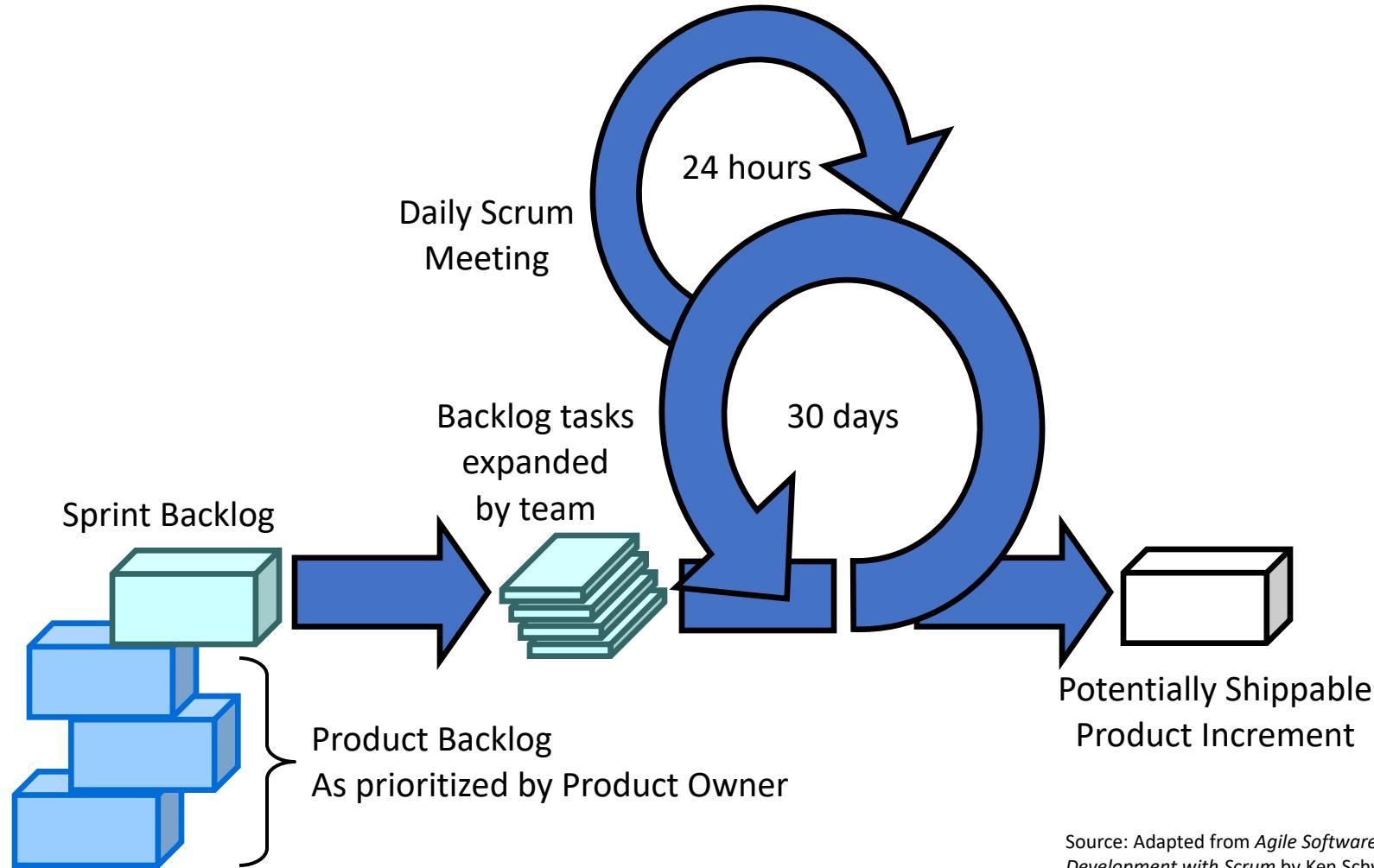
- Is an agile, **lightweight** process
- Can **manage** and **control** software and product development
- Uses iterative, incremental practices
- Has a **simple** implementation
- Increases productivity
- Reduces **time to benefits**
- Embraces **adaptive**, empirical systems development
- Is not restricted to software development projects
- Embraces the **opposite of the waterfall** approach...



Agile Manifesto

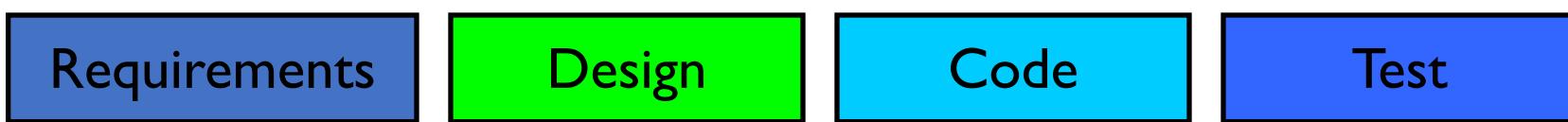


Scrum at a Glance



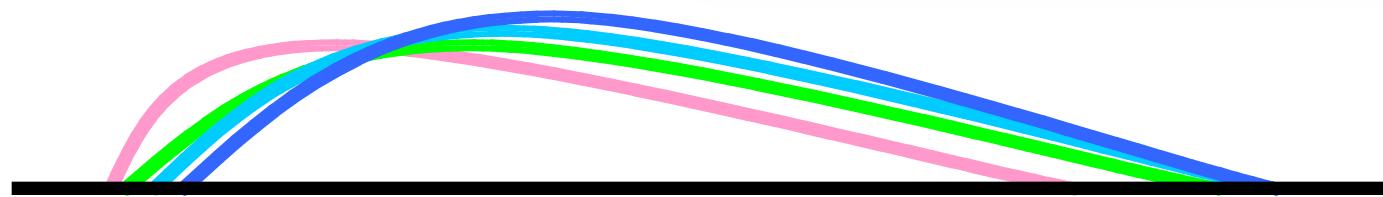
Source: Adapted from *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle.

Sequential vs. Overlap



Rather than doing all of one thing at a time...

...Scrum teams do a little of everything all the time



Scrum Framework

Roles

- Product owner
- Scrum Master
- Team

Ceremonies

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

Artifacts

- Product backlog
- Sprint backlog
- Burndown charts

Scrum Roles

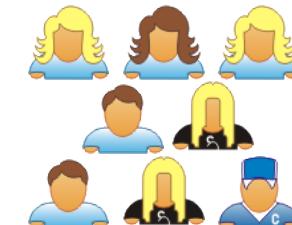
- Product Owner
 - Possibly a Product Manager or Project Sponsor
 - Decides features, release date, prioritization, \$\$\$



- Scrum Master
 - Typically a Project Manager or Team Leader
 - Responsible for enacting Scrum values and practices
 - Remove impediments / politics, keeps everyone productive



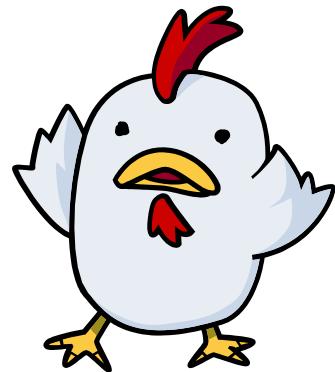
- Project Team
 - 5-10 members; Teams are self-organizing
 - Cross-functional: QA, Programmers, UI Designers, etc.
 - Membership should change only between sprints



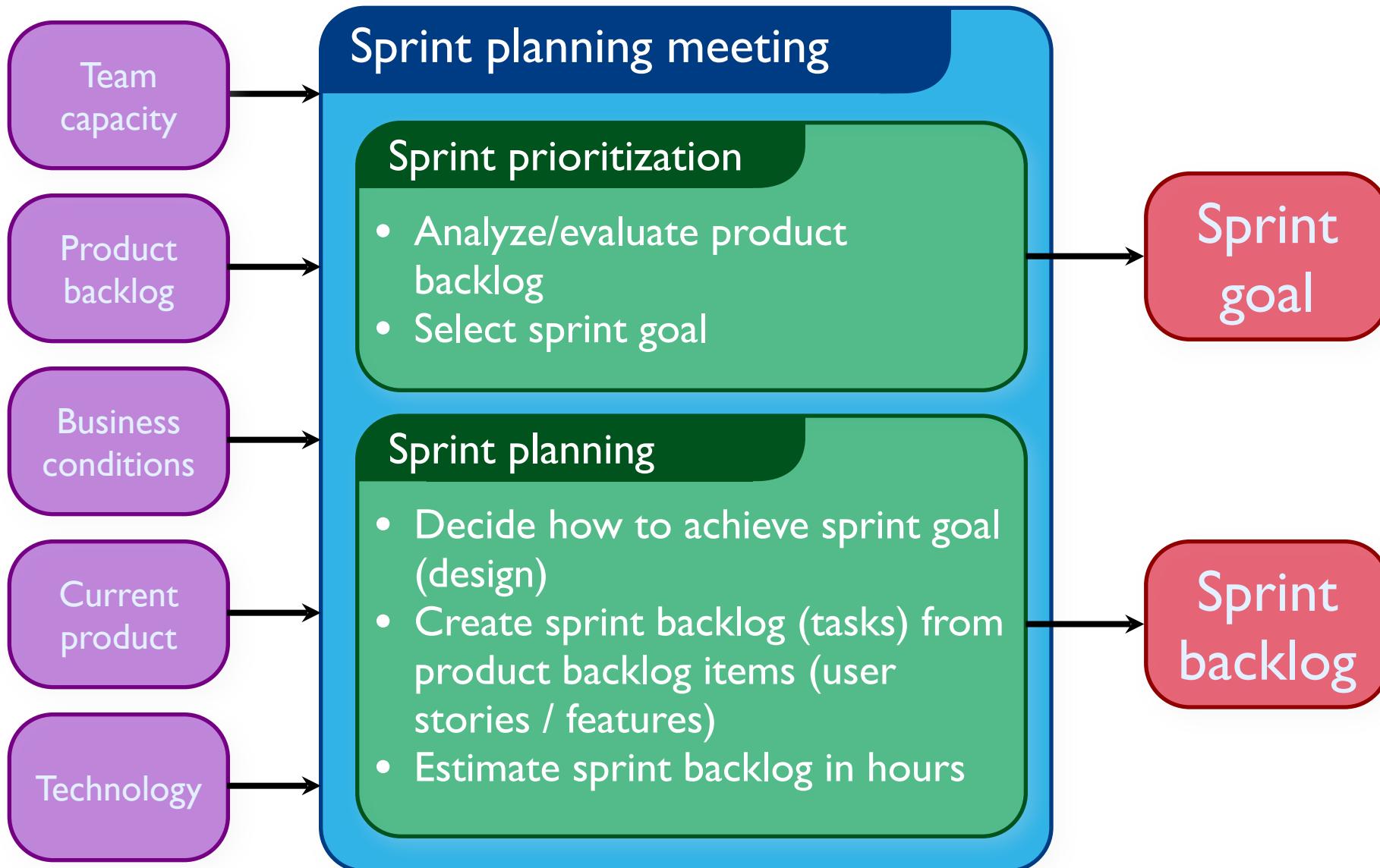
"Pigs" and "Chickens"

- **Pig:** Team member committed to success of project
- **Chicken:** Not a pig; interested but not committed

A pig and a chicken are walking down a road. The chicken looks at the pig and says, "Hey, why don't we open a restaurant?" The pig looks back at the chicken and says, "Good idea, what do you want to call it?" The chicken thinks about it and says, "Why don't we call it 'Ham and Eggs'?" "I don't think so," says the pig, "I'd be committed but you'd only be involved."



Sprint Planning Mtg.



Daily Scrum Meeting

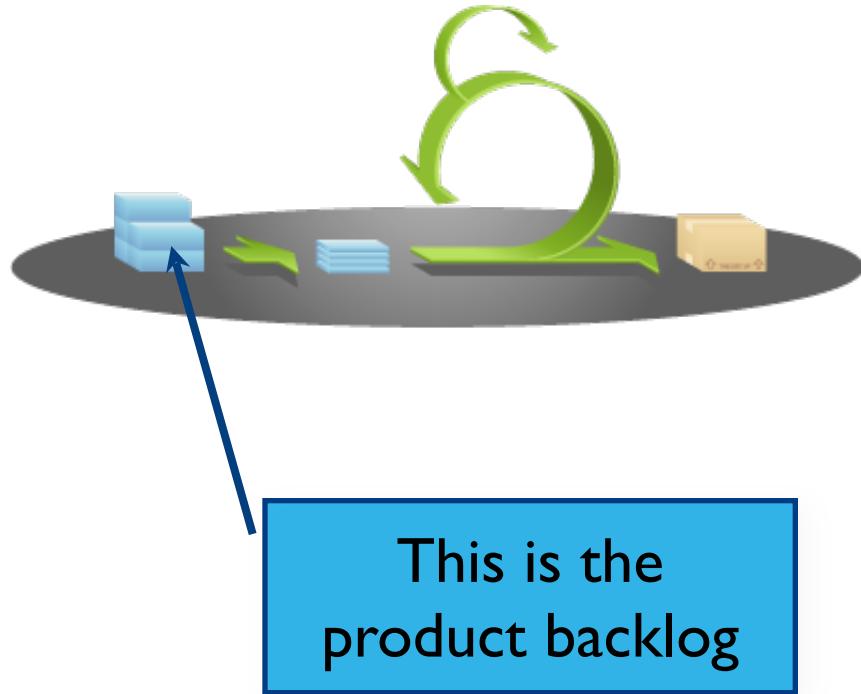
- Parameters
 - Daily, ~15 minutes, Stand-up
 - Anyone late pays a \$1 fee
- Not for problem solving
 - Whole world is invited
 - Only team members, Scrum Master, product owner, can talk
 - Helps avoid other unnecessary meetings
- Three questions answered by each team member:
 1. What did you do yesterday?
 2. What will you do today?
 3. What obstacles are in your way?



Scrum's Artifacts

- Scrum has remarkably few artifacts
 - Product Backlog
 - Sprint Backlog
 - Burndown Charts
- Can be managed using just an Excel spreadsheet
 - More advanced / complicated tools exist:
 - Expensive
 - Web-based – no good for Scrum Master/project manager who travels
 - Still under development

Product Backlog



- The requirements
- A list of all desired work on project
- Ideally expressed as a list of user stories along with "story points", such that each item has value to users or customers of the product
- Prioritized by the product owner
- Reprioritized at start of each sprint

User Stories

- Instead of Use Cases, Agile project owners do "user stories"
 - **Who** (user role) – Is this a customer, employee, admin, etc.?
 - **What** (goal) – What functionality must be achieved/developed?
 - **Why** (reason) – Why does user want to accomplish this goal?

As a [user role], I want to [goal], so I can [reason].

- Example:
 - "As a user, I want to log in, so I can access subscriber content."
- **story points:** Rating of effort needed to implement this story
 - common scales: 1-10, shirt sizes (XS, S, M, L, XL), etc.

Sample Product Backlog

| Backlog item | Estimate |
|----------------------------------------------------------------------------|------------------|
| Allow a guest to make a reservation | 3 (story points) |
| As a guest, I want to cancel a reservation. | 5 |
| As a guest, I want to change the dates of a reservation. | 3 |
| As a hotel employee, I can run RevPAR reports (revenue-per-available-room) | 8 |
| Improve exception handling | 8 |
| ... | 30 |
| ... | 50 |

Sprint Backlog

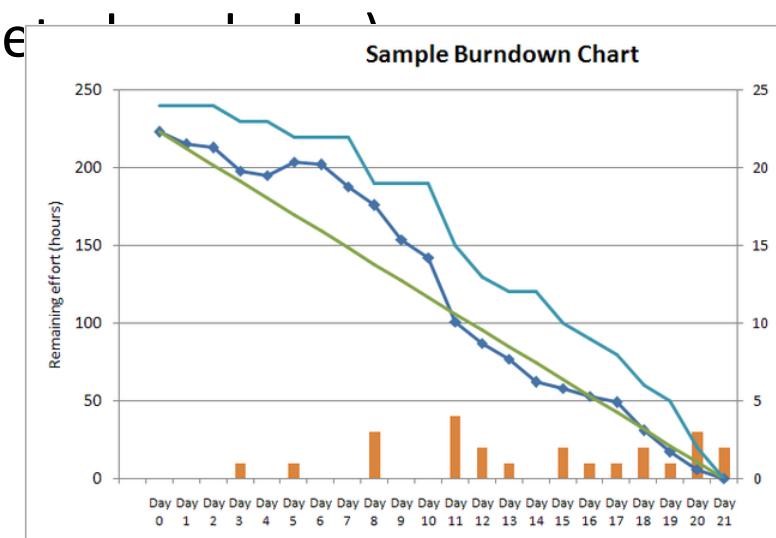
- Individuals sign up for work of their own choosing
 - Work is never assigned
 - Estimated work remaining is updated daily
-
- Any team member can add, delete change sprint backlog
 - Work for the sprint emerges
 - If work is unclear, define a sprint backlog item with a larger amount of time and break it down later
 - Update work remaining as more becomes known

Sample Sprint backlog

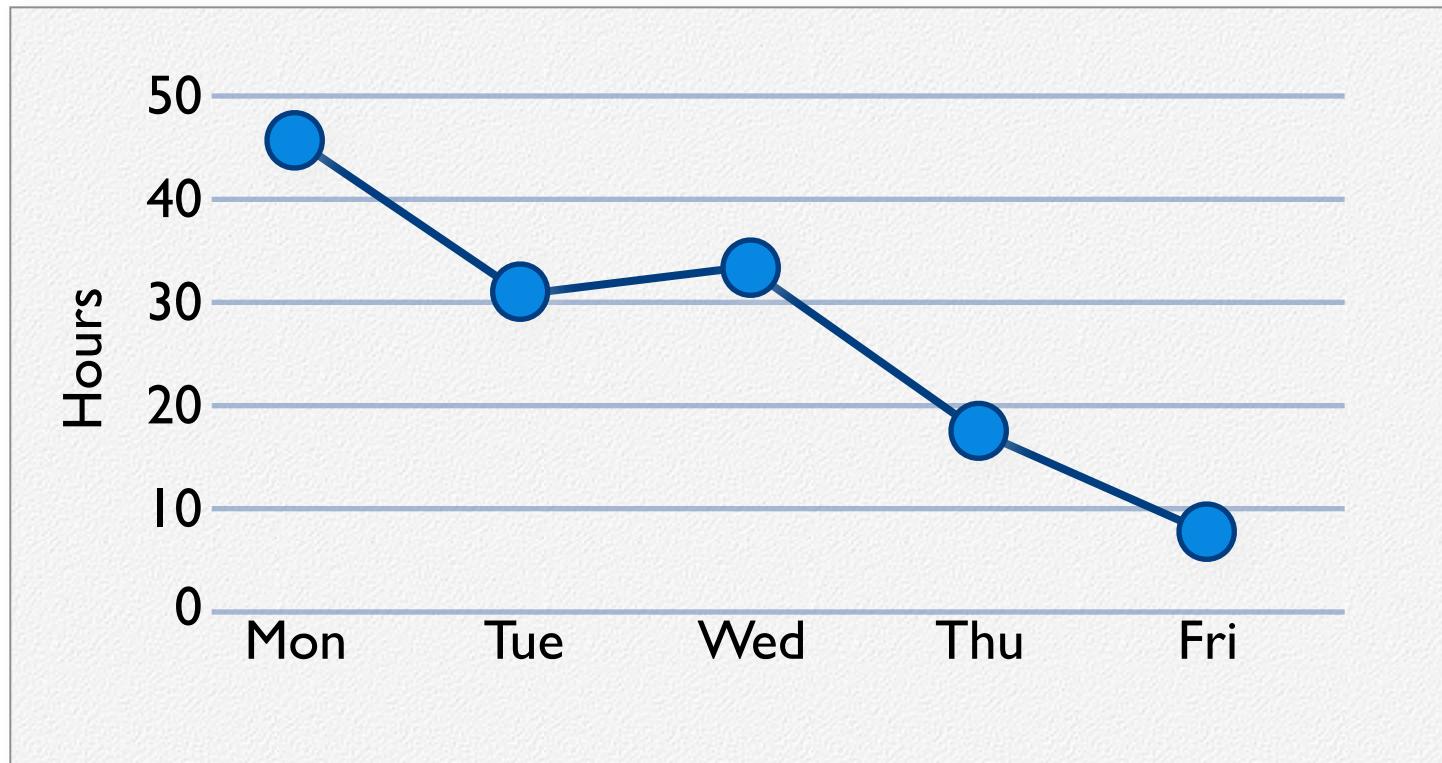
| Tasks | Mon | Tue | Wed | Thu | Fri |
|-------------------------|-----|-----|-----|-----|-----|
| Code the user interface | 8 | 4 | 8 | | |
| Code the middle tier | 16 | 12 | 10 | 4 | |
| Test the middle tier | 8 | 16 | 16 | 11 | 8 |
| Write online help | 12 | | | | |
| Write the Foo class | 8 | 8 | 8 | 8 | 8 |
| Add error logging | | | 8 | 4 | |

Sprint Burndown Chart

- A display of what work has been completed and what is left to complete
 - one for each developer or work item
 - updated every day
 - (make best guess about hours/points completed)
- *variation:* Release burndown chart
 - shows overall progress
 - updated at end of each sprint

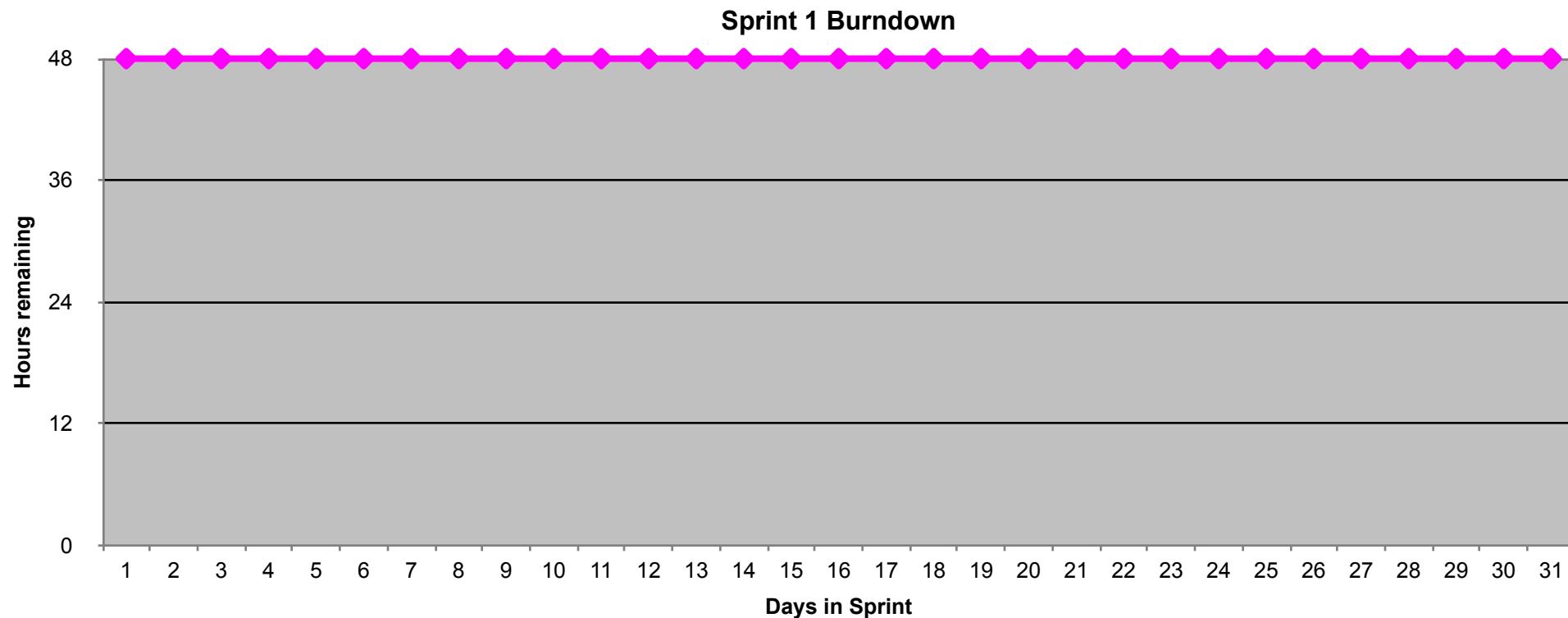


| Tasks | Mon | Tue | Wed | Thu | Fri |
|-------------------------|-----|-----|-----|-----|-----|
| Code the user interface | 8 | 4 | 8 | | |
| Code the middle tier | 16 | 12 | 10 | 7 | |
| Test the middle tier | 8 | 16 | 16 | 11 | 8 |
| Write online help | 12 | | | | |



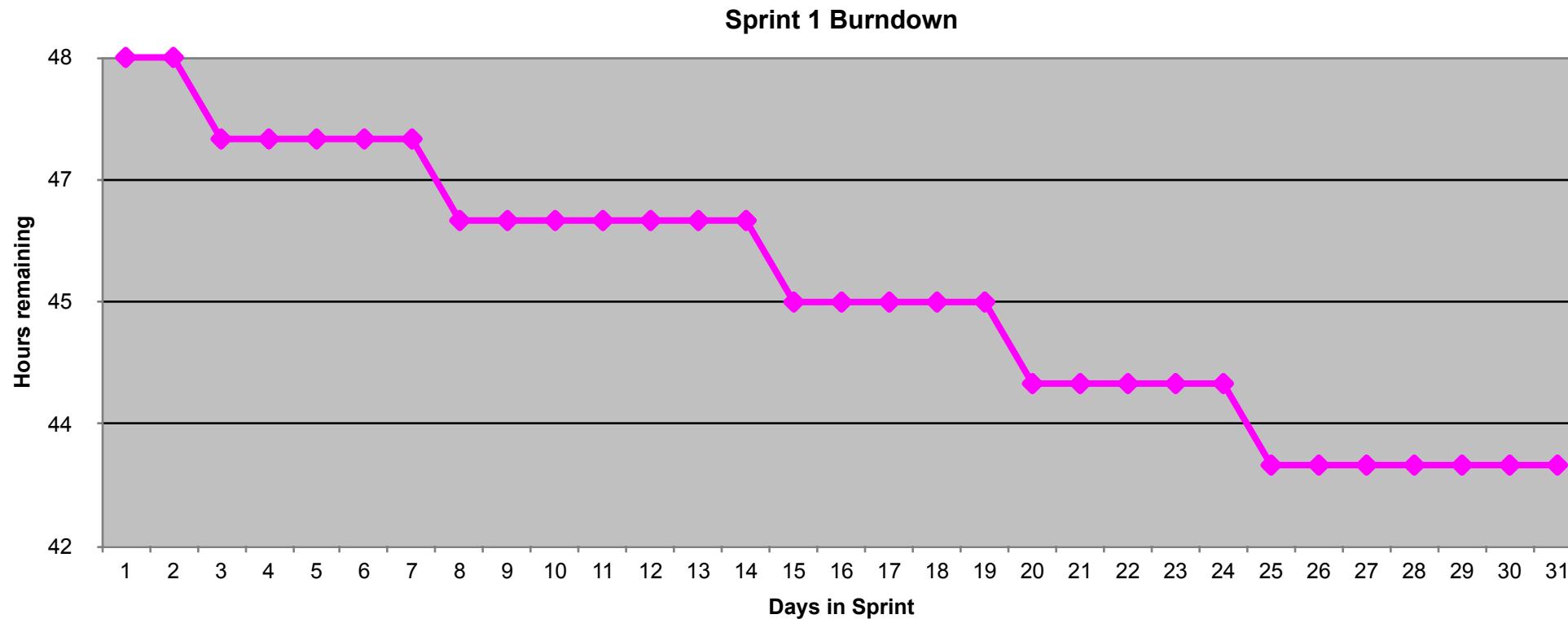
Burndown Example 1

No work being performed



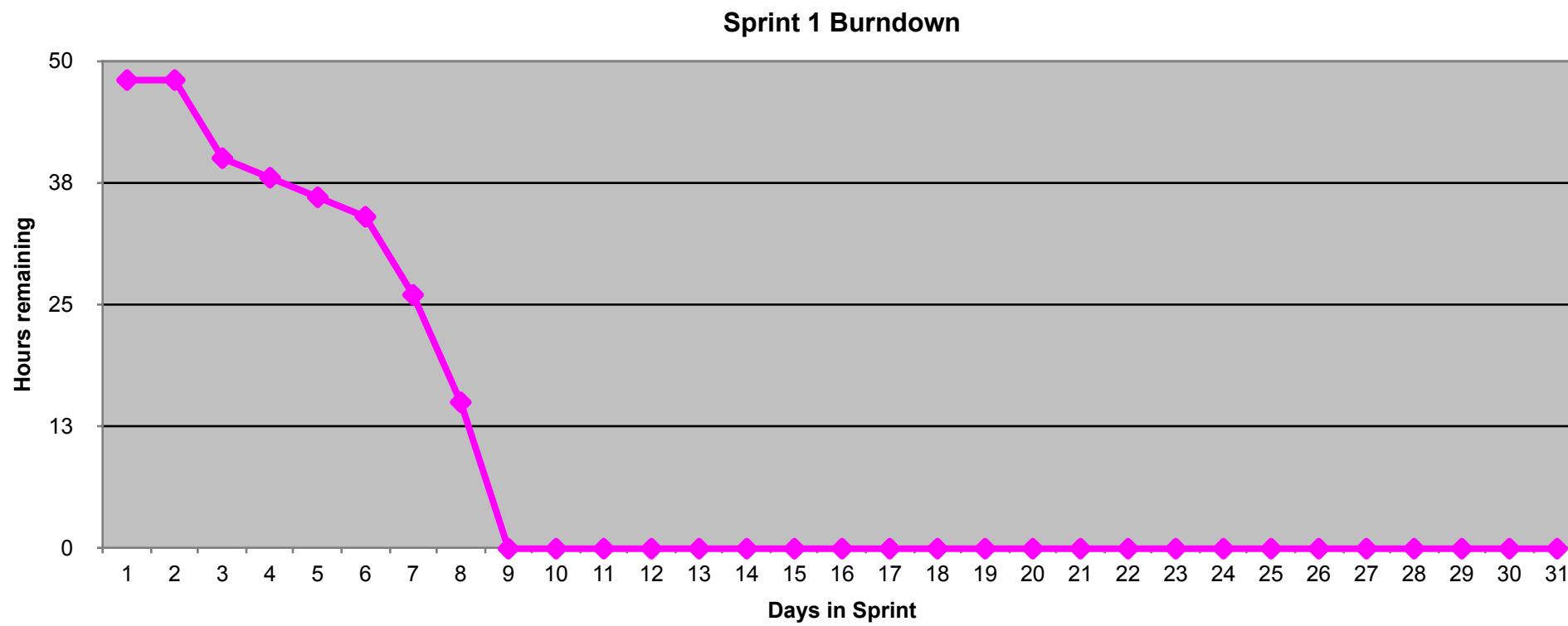
Burndown Example 2

Work being performed, but not fast enough



Burndown Example 3

Work being performed, but too fast!



The Sprint Review

- Team presents what it accomplished during the sprint
- Typically takes the form of a demo of new features or underlying architecture
- Informal
 - 2-hour prep time rule
 - No slides
- Whole team participates
- Invite the world

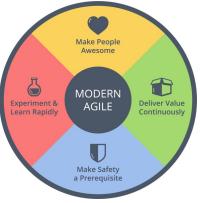


Scalability

- Typical individual team is 7 ± 2 people
 - Scalability comes from teams of teams
- Factors in scaling
 - Type of application
 - Team size
 - Team dispersion
 - Project duration
- Scrum has been used on multiple 500+ person projects

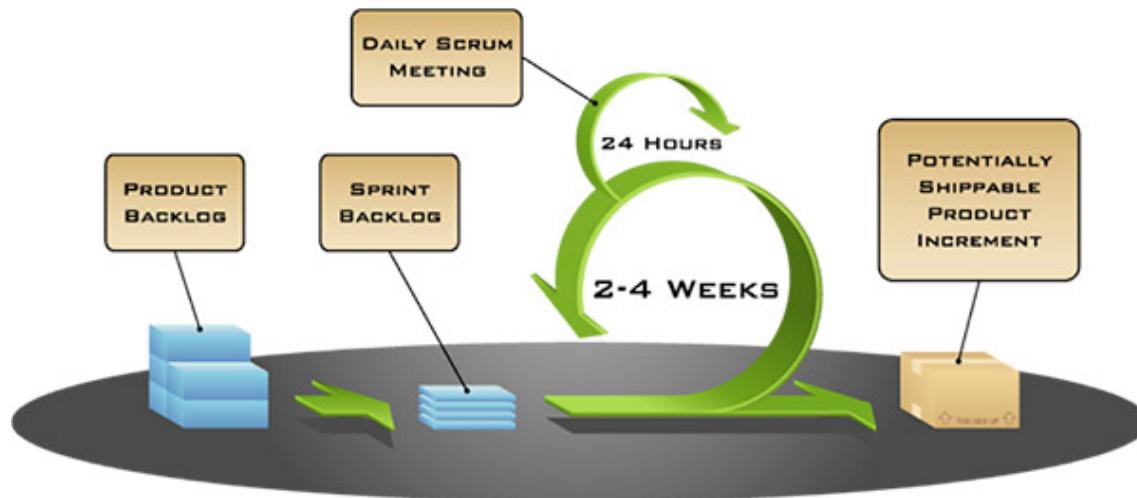
More about AGILE

Modern AGILE, AGILE in other environments

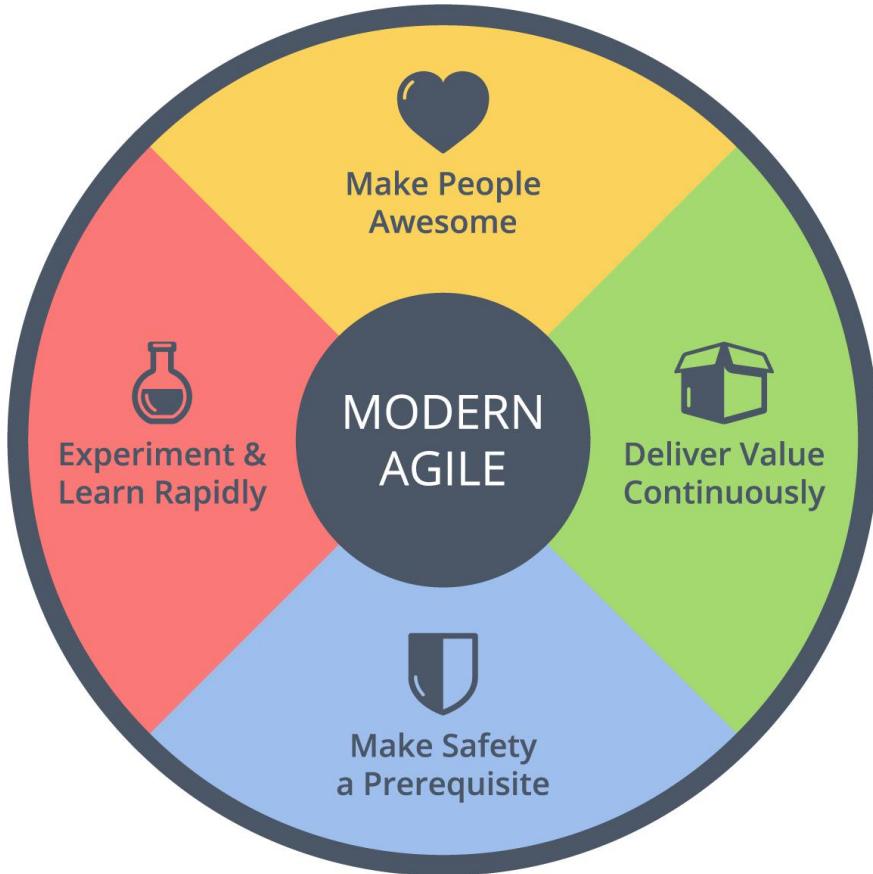


What is Modern Agile?

The industry has become overloaded with techniques, processes, methodologies and tools that have fallen under the Agile umbrella. Many consider these more marketing hype than Agile, feeling they are moving away from the simplicity that Agile principles promote. Modern Agile aims to move Agile to the next level, not by adding complexity, but by simplifying—stressing only adherence to four principles



Modern Agile Guiding Principles



Experiment & Learn Rapidly

Is a guiding principle of Modern Agile because it protects us from wasting time and helps us discover success faster

Make Safety a Prerequisite

Means establishing safety before engaging in potentially hazardous work

“Make People Awesome”
Amazon has made Customer Obsession a guiding principle since 1997 and it shows. If you make customers awesome, they tend to be natural promoters of your products or services

Deliver Value Continuously
Anything valuable that hasn't been delivered isn't helping anyone. How might we deliver the right outcomes faster

Problems with agile methods

- ◊ It can be difficult to keep the interest of **customers / users** who are involved in the process.
- ◊ Team members may be unsuited to the **intense involvement** that characterizes agile methods.
- ◊ **Prioritizing** changes can be difficult where there are **multiple stakeholders**.
- ◊ Maintaining **simplicity requires extra work**.
- ◊ **Contracts** may be a problem as with other approaches to iterative development.
- ◊ Because of their focus on small, tightly-integrated teams, there are problems in **scaling** agile methods to **large systems**.
- ◊ Less emphasis on **documentation** - harder to maintain when you get a new team for maintenance

Project Schedule / Cost Estimation Techniques

- Waterfall approach:

Bottom-Up: Detail out all requirements and estimate each task to complete those requirements in hours/days, then use this data to develop the project schedule. In the software industry, the use of the bottom-up method has severe drawbacks due to today's speed of change. *Speed of change* means that the speed of new development tools and the speed of access to new knowledge is so great that any delay in delivery leaves one open to competitive alternatives and in danger of delivering an obsolete product

- Agile approach:

Top-Down: The top-down method addresses this key issue, by using the information currently available to provide gross-level estimates. Rolling-wave planning is then used to incorporate new information as it's learned, further refining estimates and iteratively elaborating with more detail as the project progresses. This method of learning just enough to get started, with a plan to incorporate more knowledge as work outputs evolve, allows the project team to react quickly to adversity and changing market demand

(source: PMI)



How to estimate project schedule in Agile?

- Determine **Point Value** for each item to be worked on. The most popular technique of gross level estimation is **Planning Poker**, using Fibonacci sequence to assign a point value to a feature or item
- Determine **Team Velocity**. A team's average velocity is used in forecasting a long-term schedule. Average velocity is calculated by summing and averaging the velocity measurements from the team's last three iterations
- The team's average velocity number is used to calculate the most likely scenario, while velocity numbers from the team's worst-performing iterations are used to calculate the most **pessimistic forecast** completion date. Using velocity from iterations where the team was able to complete more than expected provides the most **optimistic forecast**



How to estimate project cost in Agile?

- A simple formula is used to determine the cost per point:

$$\Sigma (\text{loaded team salaries for period } n) / \text{points completed in period } n$$

A team whose total loaded salaries are \$240,000 over six weeks, and completed 60 points of work in those three iterations, would have a cost per point of \$4,000

- Use the following formula to determine budget:

$$(\text{Cost per point} \times \text{total point value of items to be completed}) + \text{other expenses} = \text{forecast budget}$$

Budget estimates are based on what we know today

In the example used, budget estimate is for the first release and not the entire project. The team could apply an additional 20% for the second release and an additional 5% for the last release, based on past experience

What is DevOps?

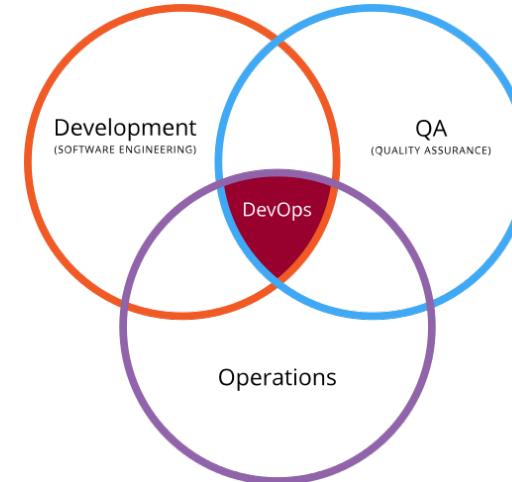
We are now on Agile bandwagon (hooray)! Development teams are working on the software in short sprints lasting not more than two weeks.

Having such a short release cycle helped the development team work on client feedback and incorporate it along with bug fixes in the next release.

While this Agile SCRUM approach brought agility to development, it was lost on Operations which did not come up to speed with Agile practices. Lack of collaboration between Developers and Operations Engineers still slowed down the development process and releases.

DevOps Methodology was born out of the need for better collaboration and faster delivery. DevOps enables continuous software delivery with less complex problems to fix and faster resolution of problems.

In summary: **Agile** is a set of values and principles about how to develop software. If you have ideas and you want to turn those ideas into working software, you can use the Agile values and principles as a way to do that. But, that software might only be working on a developer's laptop or in a test environment. You want a way to quickly, easily and at will move that software into production infrastructure, in a safe and simple way. To do that you need **DevOps** tools and techniques.



Venn Diagram

Does Agile work for infrastructure projects?



Agile works very well in projects where more is unknown than known. This is common in software development projects.

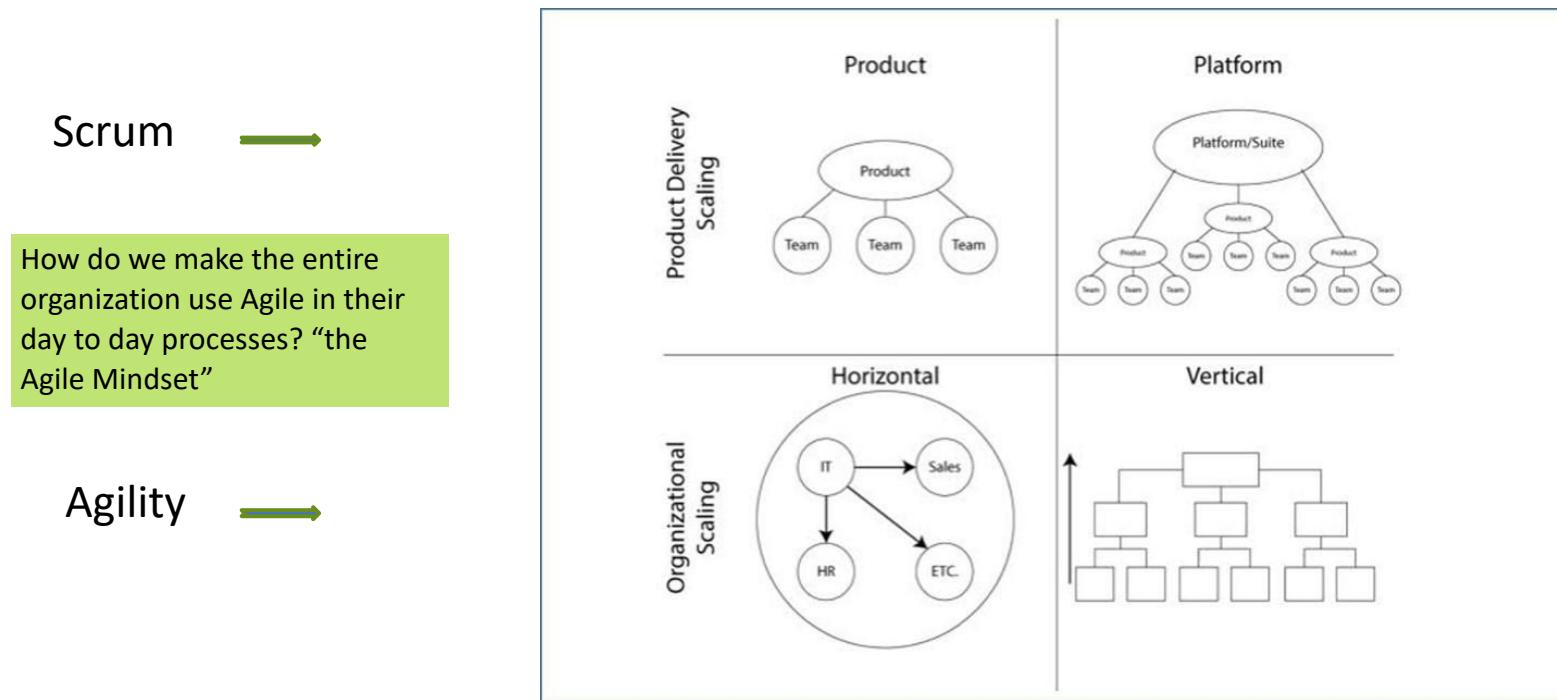
In software development end users kind of know what they want. The requirements evolve over time. For example: end user thinks they need some kind of a shape. You start with something like circle. After an iteration the requirement might change from circle to square. After another iteration it might change from square to square filled with some color. The requirements evolve. Agile methodology has framework to support "**unknown**" throughout the project life cycle.



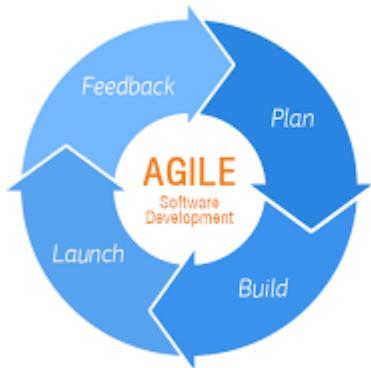
In contrast infrastructure projects have very well defined requirements (very few unknowns), strict dependency between tasks, mainly configuration driven, some tasks are done only during permitted windows (weekends or after business hours). For example **storage expansion** tasks are comprised of procurement, cabling, network and storage tasks. These tasks have dependency and follow one after another. Projects like this are well suited for waterfall methodology. But infrastructure projects which have "**unknowns**" and similar to software development (BaaS, IaaS, PaaS) can greatly benefit from agile methodology

What is Scaling Framework?

Scaling is the application of practices of “Agility” across “Horizontal” and “Vertical” space through an organization. Scrum (a tool used to organize work into small, manageable pieces that can be completed by a cross-functional team within a prescribed time period) is a process unique to software development and can not be applied outside of that paradigm. Agile mindset and Lean thinking however can be practiced across all organizational teams.



SAFe: Agile Software Development



Agile software development refers to a group of software development methodologies (*Scrum, Kanban, and Extreme Programming*) based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams

Agile development refers to any development process that is aligned with the concepts of the Agile Manifesto. The Manifesto was developed by a group leading figures in the software industry, and reflects their experience of what approaches do and do not work for software development

SAFe: Lean Software Development Principles

Lean development can be summarized by seven principles, very close in concept to lean manufacturing principles:

- Eliminate Waste
- Amplify Learning
- Decide as late as possible
- Deliver as fast as possible
- Empower the team
- Build integrity in
- See the whole

SAFe: What is Systems Thinking

Systems Thinking is a management discipline that views the complete organization in relation to its environment. It provides a means of understanding, analyzing and talking about the design and construction of the organization as an integrated, complex composition of many interconnected systems (human and non-human) that need to work together for the whole to function successfully.



What is SAFe?

The **Scaled Agile Framework (SAFe)**, is intended to guide enterprises in scaling lean and agile practices. SAFe promotes alignment, collaboration, and delivery across large numbers of agile teams (Agile on steroids).

It was developed by and for practitioners, by leveraging three primary bodies of knowledge:

1. Agile software development (Agile Software Development is an umbrella term for a set of methods and practices based on the values and principles expressed in the Agile Manifesto. Solutions evolve through collaboration between self-organizing, cross-functional teams utilizing the appropriate practices for their context)
2. Lean product development (Eliminate Waste, Amplify Learning, Decide as late as possible, Deliver as fast as possible, Empower the team, Build integrity, See the whole)
3. Systems thinking (Is a management discipline that concerns an understanding of a system by examining the linkages and interactions between the components that comprise the entirety of that defined system)

Agile Release Train (ART)



- **What is it?**

The ART metaphor describes the program level teams, roles, and activities that incrementally deliver a continuous flow of value. ARTs are virtual organizations formed to span functional boundaries, eliminate unnecessary handoffs and steps, and accelerate value delivery by implementing SAFe Lean-Agile principles and practices

- **Who is responsible for it?**

System Architect/Engineer— Is an individual or small cross-discipline team that truly applies Systems Thinking. They define the overall architecture for the system, help define Nonfunctional Requirements (NFRs), determine the major elements and subsystems, and help design the interfaces and collaborations among them.

Product Management – Is the internal voice of the Customer and works with customers and Product Owners to understand and communicate their needs, define system features, and participate in validation. They are responsible for the Program Backlog.

Release Train Engineer (RTE) – Is a servant leader and the chief Scrum Master for the train. The RTE facilitates optimizing the flow of value through the program using various mechanisms, such as the Program Kanban, Inspect & Adapt (I&A) workshop, and PI Planning.

Business Owners – Are a small group of stakeholders who have the business and technical responsibility for fitness for use, governance, and return on investment (ROI) for a Solution developed by an ART. They are primary stakeholders in the ART and actively participate in ART events

Should a Project Manager learn Agile?

The impact of Agile on the role of many project managers is likely to be significant. On small, single-team Agile projects, you may not find someone called a “Project Manager”. However:

- ✓ There certainly is a need for someone to coach and mentor the team on Agile Project Management practices.
- ✓ There is also a need for project managers on larger and more complex enterprise-level projects but even at that level, some understanding of Agile is essential.

This “raises the bar” for project managers significantly. It requires project managers to develop a fresh new perspective to see Agile and traditional, plan-driven project management approaches as complementary to each other rather than competitive and to learn how to blend the two approaches in whatever proportions are needed to fit any situation

The PM role in a Lean and Agile world

In the lean and agile world, the project manager does not have an official role

The Scrum practice prescribes distributing the PM role among the Scrum team members. However, there are varying opinions and experiences assigning any combination of the 3 roles to a PM.

1. **The Scrum Master** (The *Scrum Master* does anything possible to help the team perform at their highest level. This involves removing any impediments to progress, facilitating meetings, and doing things like working with the product owner to make sure the product backlog is in good shape and ready for the next sprint)
2. **The product Owner** (The *Product Owner* is typically a project's key stakeholder. Part of the product owner responsibilities is to have a vision of what he or she wishes to build, and convey that vision to the scrum team. This is key to successfully starting any agile software development project. The agile product owner does this in part through the product backlog, which is a prioritized features list for the product)
3. **The development team member** (Responsible for the project's creation and delivery. The team members will normally be comprised of developers, QA, and documentation. They are responsible for planning, design, development, testing, and project delivery)

The Scaled Agile Framework (SAFe) practice lists the PM as a potential for the Release Train Engineer (RTE). Other agile practitioners describe the PM as a coach and facilitator

Project Manager as a ScrumMaster

- ❑ **Authority:** The traditional project manager role moves from a command-and-control, hierarchical position to a servant-leader or facilitator position
- ❑ **Requirements:** The product owner assumes the responsibility for ensuring the requirements are defined
- ❑ **Work Assignments:** The team takes ownership and accountability for meeting the project and team goals
- ❑ **Managing Stakeholder Expectations:** Product owner provides direction and leadership to the team
- ❑ **Leadership and Support:** The Scrum Master serves the product owner and the team so that they are better able to do their jobs by assisting them, facilitating creativity and fostering empowerment
- ❑ **Removes Impediments:** The Scrum Master helps remove obstacles and support the team

The



- A project manager (PM) is a highly skilled knowledge worker who has received rigorous training and knowledge in the process of achieving a globally recognized certification
- The SAFe model represents enterprise agility and extends Scrum beyond the team execution level into the organization. The PM role is best characterized through enterprise agility and viewed through the lens of portfolio, program, and the execution teams that are aligned to ensure maximum customer value
- The knowledge and skills obtained through certification is transferable in the lean and agile organization. In a competitive business climate, all available brainpower must be present on deck to enable the organization to achieve **enterprise** agility and scale to meet customer, compliance, financial markets, internal opportunities, and competitive demands

Agile Certified Professional

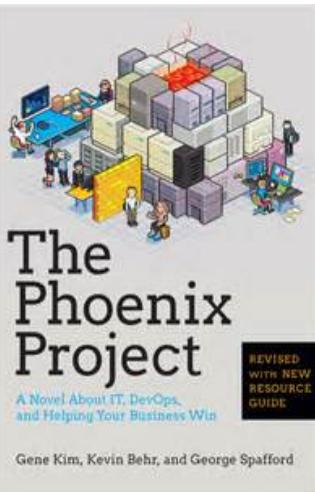
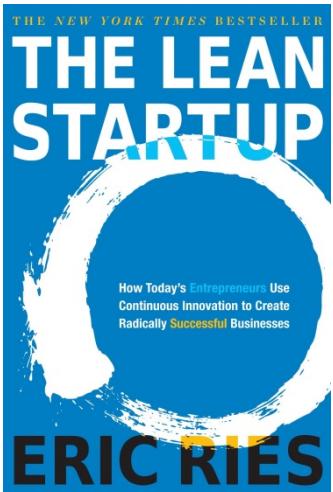
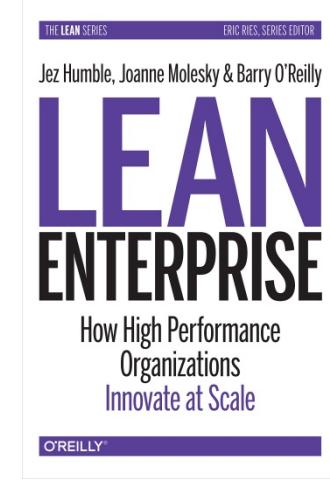
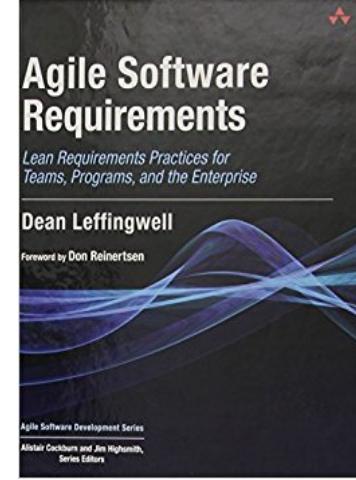
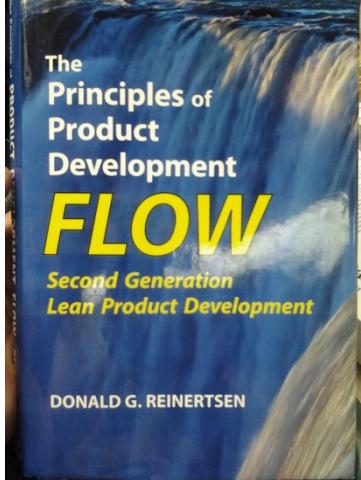
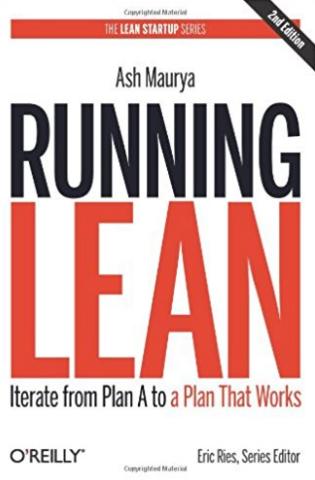
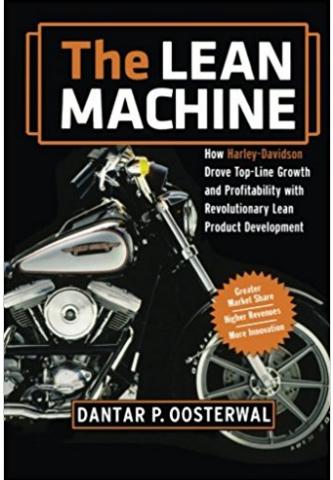
Become a PMI Agile Certified Professional (PMI-ACP):

| General Project Experience | Project Experience | Training | Examination |
|-------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------|----------------------|--------------------------|
| 2,000 hours working on project teams within the last five years. The PMP certification satisfies this requirement | 1,500 hours working on agile projects within the last three years | 21 hours of training | Pass a 200-question test |

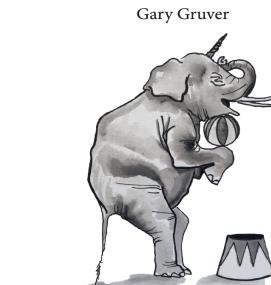
The PMI Agile Certified Practitioner (PMI-ACP)® formally recognizes your knowledge of agile principles and your skill with agile techniques. It will make you shine even brighter to your employers, stakeholders and peers

The PMI-ACP spans many approaches to agile such as Scrum, Kanban, Lean, extreme programming (XP) and test-driven development (TDD.) It will increase your versatility, wherever your projects may take you.

Recommended Reading



Starting and Scaling DevOps in the Enterprise



"With all of the hype around DevOps, it has been difficult to really understand how DevOps scales at large companies and how to begin. The insight in this book has provided clarity and a path forward. I look forward to beginning the process using this book as the guide."

- Steven D. Leist, VP, American Airlines



ALEX YAKYMA

Recommended Videos

- ❑ Daniel Pink Drive - <https://www.youtube.com/watch?v=uwA97yWz9Uc>
- ❑ Nordstrom Innovation Lab - <https://www.youtube.com/watch?v=szr0ezLyQHY>
- ❑ Agile Product Owner in a Nutshell - <https://www.youtube.com/watch?v=502ILHjX9EE>
- ❑ John Kotter Our Iceberg is Melting Video - <https://www.youtube.com/watch?v=Gh2xc6vXQgk>
- ❑ Submarine Captain - David Marquet - https://www.youtube.com/watch?v=OqmdLcyES_Q
- ❑ Servant Leadership - <https://www.youtube.com/watch?v=aKk0AaaFqtU>
- ❑ Spotify Engineering Culture Part 1 - <https://www.youtube.com/watch?v=4GK1NDTWbkY>
- ❑ Spotify Engineering Culture Part 2 - <https://www.youtube.com/watch?v=X3rGdmoTjDc>
- ❑ Radical Candor The Surprising Secret to Being a Good Boss - <https://www.youtube.com/watch?v=4yODalLQ2IM>
- ❑ How Business Stakeholders Work With Agile Teams - <https://www.stickystories.co/whats-different-for-business-stakeholders-when-its-an-agile-approach/>
- ❑ The Backwards Brain Bicycle - Smarter Every Day 133 - <https://www.youtube.com/watch?v=MFzDaBzBILO>

Glossary of Terms

| | |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Agile Manifesto | Is a formal proclamation of 4 key values and 12 principles to guide an iterative and people-centric approach to software development. |
| ART | Agile Release Train - Is a virtual organization (50 – 125 people) that plans, commits, and executes together |
| BaaS | Backend as a Service |
| Backlog | A backlog is an ordered list of items representing everything that may be needed to deliver a specific outcome. There are different types of backlogs depending on the type of item they contain and the approach being used |
| Business Agility | Business agility is the ability of an organization to sense changes internally or externally and respond accordingly in order to deliver value to its customers |
| CoP (in SAFe) | Culture built on professional networking, personal relationships, shared knowledge, and common skills. Combined with voluntary participation, CoPs provide knowledge workers with opportunities to experience autonomy, mastery, and purpose beyond their daily tasks on an Agile Release Train (ART) |
| Cross Functional | Relating to a system whereby people from different areas of an organization work together as a team |
| DoD | Definition of Done |
| Enabler | Enablers support the activities needed to support efficient development and delivery of future business requirements |
| Epic | An epic is a large user story |
| Extreme Programming | Extreme Programming (XP) is an agile software development framework that aims to produce higher quality software, and higher quality of life for the development team. XP is the most specific of the agile frameworks regarding appropriate engineering practices for software development |
| IaaS | Infrastructure as a Service |
| Iteration | An iteration is a timebox during which development takes place. The duration may vary from project to project and is usually fixed |
| KPI | Key Performance Indicators – Example: KPIs of a shoe factory, would be number of shoes <i>without defects</i> made in a period of time |

Glossary of Terms continued..

| | |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Mob Programming | Mob Programming is a software development approach where the whole team works on the same thing, at the same time, in the same space, and at the same computer |
| NFR | Non-Functional Requirements (security, reliability, performance, maintainability, scalability) |
| PaaS | Platform as a Service (Cloud) |
| Pair Programming | Is an agile software development technique in which two programmers work together at one workstation. One, the driver, writes code while the other, the observer or navigator, reviews each line of code as it is typed in |
| PI | A routine, face-to-face event, with a standard agenda that includes a presentation of business context and vision followed by team planning breakouts—where the teams create their iteration plans and objectives for the upcoming PI |
| Planning Poker | An approach to estimation used by Agile teams. Each team member "plays" a card bearing a numerical value corresponding to a point estimation for a user story |
| Refactoring | Refactoring consists of improving the internal structure of an existing program's source code, while preserving its external behavior |
| Retrospective | The team meets regularly to reflect on the most significant events that occurred since the previous such meeting, and identify opportunities for improvement |
| RTE | Release Train Engineer - The RTE's major responsibilities are to facilitate the ART events and processes and assist the teams in delivering value. RTEs communicate with stakeholders, escalate impediments, help manage risk, and drive relentless improvement |
| Scrum Master | The scrum master is responsible for ensuring the team lives agile values and principles and follows the practices that the team agreed they would use |
| Scrum of Scrums | A technique to scale Scrum up to large groups (over a dozen people), consisting of dividing the groups into Agile teams of 5-10 |
| Spike | In agile software development, a spike is a story that cannot be estimated until a development team runs a time boxed investigation. The output of a spike is an estimate for the original story |
| UI | User interface (UI) is the series of screens, pages, and visual elements—like buttons and icons—that you use to interact with a device |
| UX | User experience (UX), is the internal experience that a person has as they interact with every aspect of a company's products and services |

Introduction to JIRA

AGILE Software Tools

Topics for today

- Introduction to JIRA and its features
- Demonstration of usage of JIRA
- JIRA for Scrum
- Comparing JIRA with other version control tools
- Q & A

Basic questions

- What is JIRA? In simple terms: “JIRA is an issue tracker”
- Is it web based? Yes
- Is it open source? Yes
- Is it a licensed product? Yes
- Whose product is JIRA? www.atlassian.com

Features (from the creators of JIRA)

- **Because you've got issues**

- JIRA lets you prioritise, assign, track, report and audit your 'issues,' whatever they may be — from software bugs and help-desk tickets to project tasks and change requests.

- **Reporting and statistics**

- Customisable reporting allows you to monitor the progress of your issues with detailed graphs and charts.

- **Workflow your way**

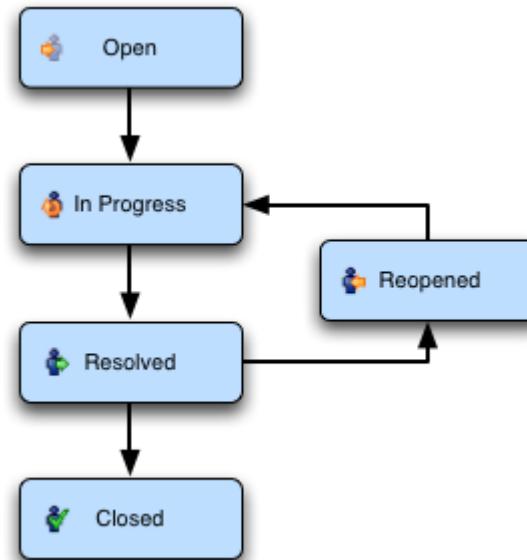
- Map your business process with a custom workflow.

- **An extensible platform**

- Integrate JIRA into your systems with our open API and 100+ free plugins.

Issues and Workflows

- What is an “Issue”?
 - Any task that requires an action from a person
- What is “Workflow”?



- What are fields?
 - The attributes of an issue (Standard & Custom)

Workflow- Scrum

- Prioritize the Backlog.
- Estimate the stories
- Create the sprints
- Story points are available only for Epics and Stories
- Estimations in terms of Hours also possible by configuring the board
- Sub tasks can be created
- The stories can be grouped under each sprint
- The number of issues and the story points / hours are automatically displayed to make plan easier
- At one point of time only one sprint is active

Show Sample Project

jira.attlasian.com

Topics

- Workflow Types
- Resolved ≠ Closed
- Do's and Don'ts
- Custom Workflow Planning
- Custom Workflow Process
- Workflow Concepts
- Add-ons & Plugins
- Resources

Workflow Types

Image: Jira Server Project Templates

Create project

SOFTWARE

- Scrum software development**
Agile development with a board, sprints and stories. Connects with source and build tools.
- Kanban software development**
Optimise development flow with a board. Connects with source and build tools.
- Basic software development**
Track development tasks and bugs. Connects with source and build tools.

BUSINESS

- Project management**
Plan, track and report on all of your work within a project.
- Task management**
Quickly organize and assign simple tasks for you and your team.
- Process management**
Track all the work activity as it transitions through a streamlined process.

Import a project | Create with shared configuration | Create sample data

Next Cancel

Image: Jira Cloud Project Templates

Choose a template

Templates allow you to quickly create a project with our recommended configuration. You can customize parts of it later.

Scrum

- Manage stories, tasks, and workflows for a scrum team
- For teams that deliver work on a regular schedule

Kanban

- Monitor work in a continuous flow for agile teams
- Suits teams who control work volume from a backlog

Bug tracking (formerly Basic software development)

- Manage a list of development tasks and bugs
- Great for teams who don't need a board

Agility

- Manage a software project with this easy, simplified board
- Suits teams that are new to agile

Project Management

- Manage activities for completing a business project
Use it for time-based or deliverable-based projects

Select Cancel

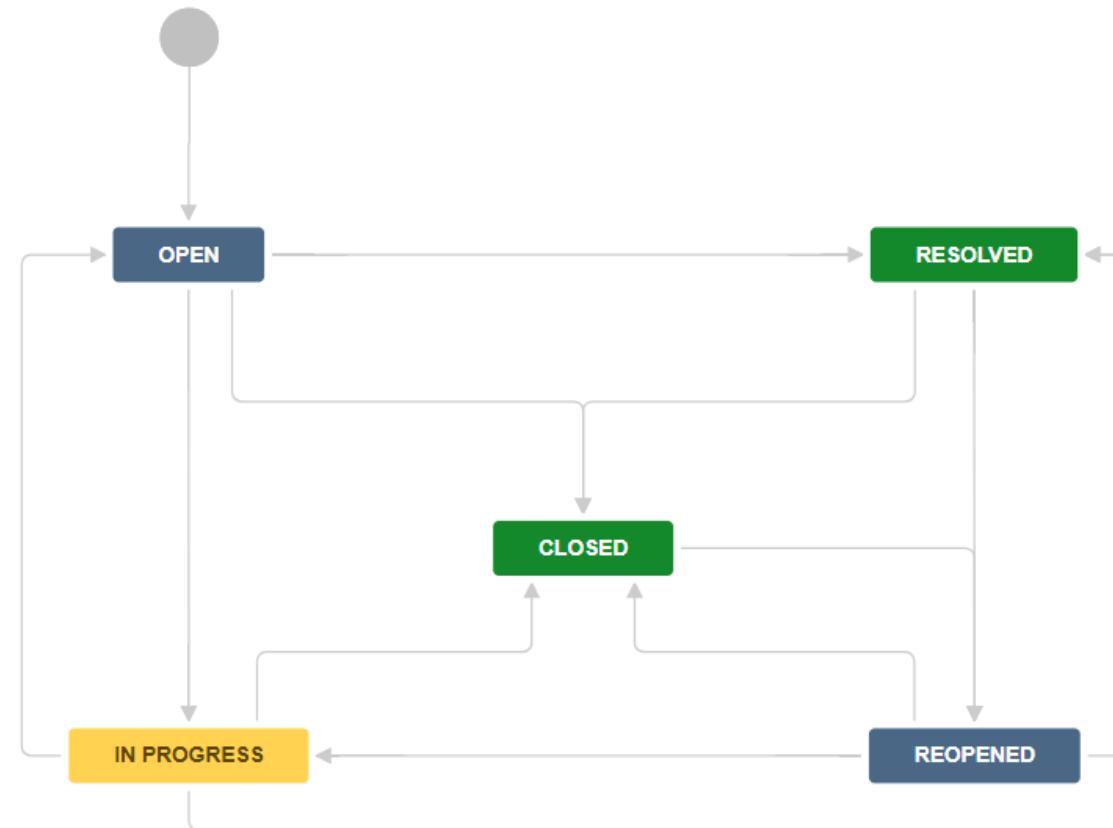
Workflow Types

Image: Admin > Issues > Workflows

| Workflows | | | | | Add workflow | Import ▾ | ? |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------|---------------------------------------------|-------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|-------------------|
| | | | | |  To delete a workflow, you must first unassign it from all workflow schemes and draft workflow schemes. | | |
|  Active | | | | | | | |
| Name | Last modified | Assigned Schemes | Steps | Actions | | | |
| Software Simplified Workflow for Project DEMO Generated by JIRA Software version 7.7.0- DAILY20180130085629. This workflow is managed internally by JIRA Software. Do not manually modify this workflow. | 15/Feb/18 rwright@jirastrategy.com | • DEMO: Software Simplified Workflow Scheme | 3 | View | Edit | Copy | |
|  Inactive | | | | | | | |
| Name | Last modified | Assigned Schemes | Steps | Actions | | | |
| jira (Read-only System Workflow) <small>DEFAULT</small> The default JIRA workflow. | | | 5 | View | Copy | | |
| classic default workflow The classic JIRA default workflow | | • classic | 5 | Edit | Copy | | |

Default Workflows

Image: Read-only System Workflow (Diagram Mode)



Default Workflows

Image: Read-only System
Workflow (Text Mode)

| Step Name (id) | Linked Status | Transitions (id) |
|-----------------|---------------|-------------------------------------------------------------------------------------------------------------------------------|
| Open (1) | OPEN | <i>Start Progress</i> (4) >> IN PROGRESS <i>Resolve Issue</i> (5) >> RESOLVED <i>Close Issue</i> (2) >> CLOSED |
| In Progress (3) | IN PROGRESS | <i>Stop Progress</i> (301) >> OPEN <i>Resolve Issue</i> (5) >> RESOLVED <i>Close Issue</i> (2) >> CLOSED |
| Resolved (4) | RESOLVED | <i>Close Issue</i> (701) >> CLOSED <i>Reopen Issue</i> (3) >> REOPENED |
| Reopened (5) | REOPENED | <i>Resolve Issue</i> (5) >> RESOLVED <i>Close Issue</i> (2) >> CLOSED <i>Start Progress</i> (4) >> IN PROGRESS |
| Closed (6) | CLOSED | <i>Reopen Issue</i> (3) >> REOPENED |

Software vs. Business Workflows

Software

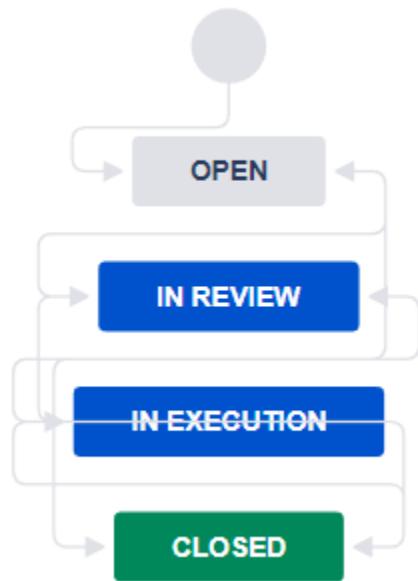
- Receive Request
- Review Requirements
- Write Code
- Test Code
- Deploy Code
- etc.

Business

- Extra Steps
- Forward & Backward Movement
- Conditional
- Highly Customized
- etc.

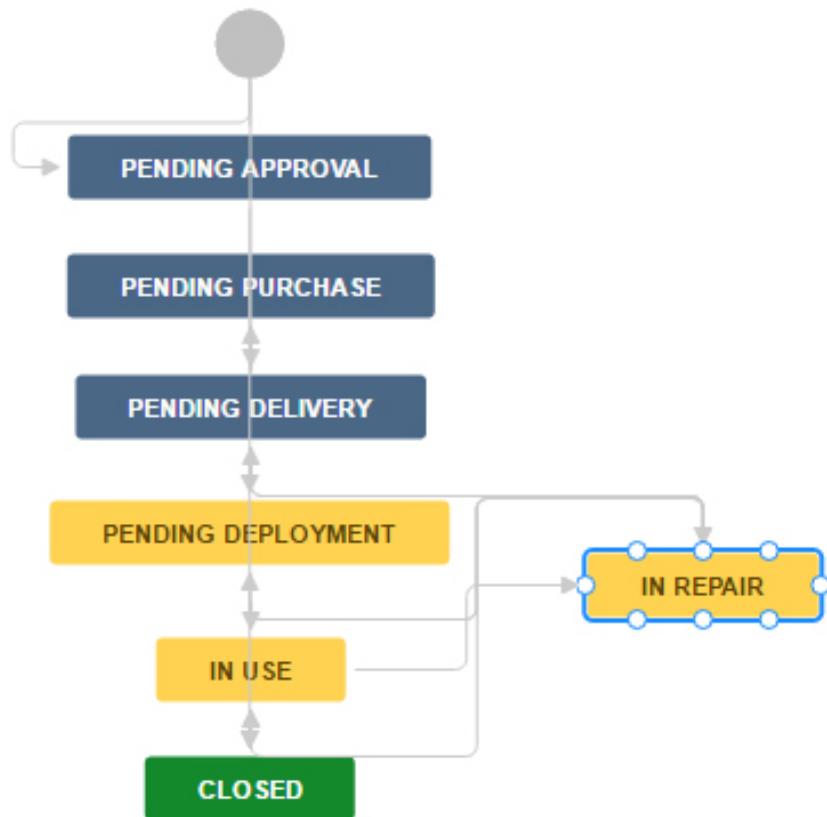
Sample Workflow

Image: Legal Contract Workflow (Jira Cloud)



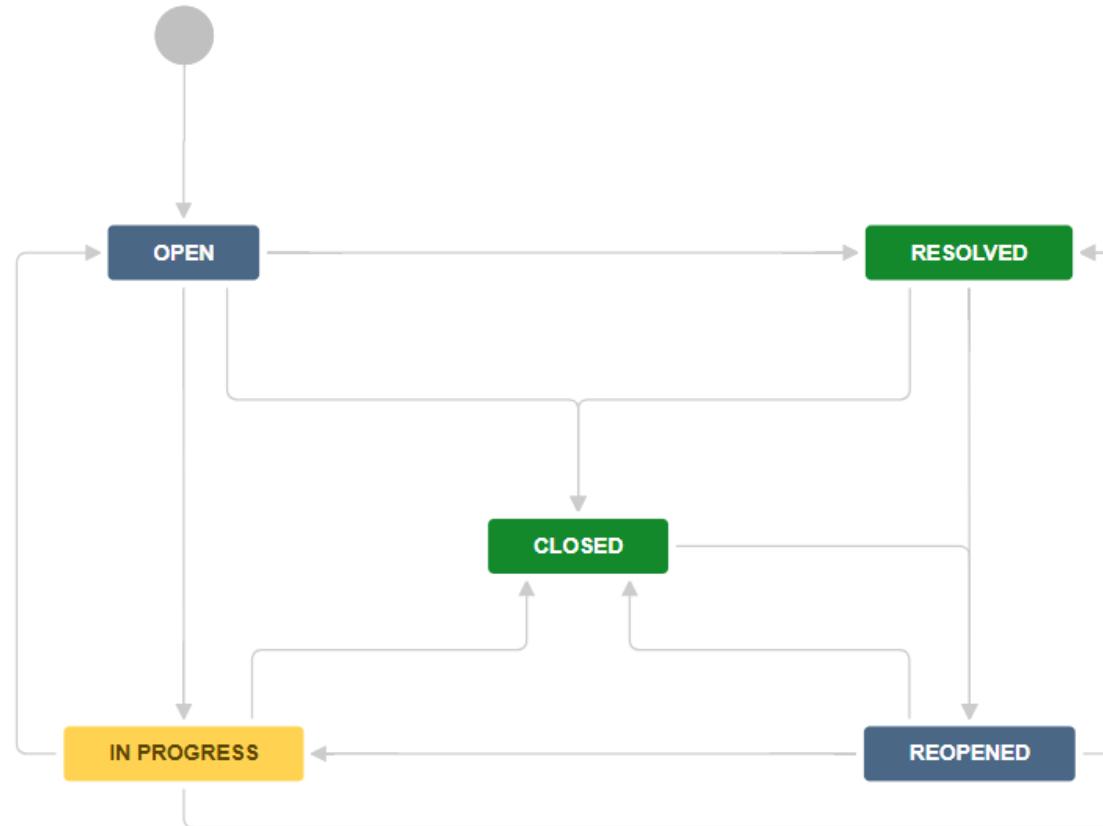
Sample Workflow

Image: Asset Management Workflow (Jira Server)



Resolved ≠ Closed

Image: Default Workflow



Do's and Don'ts

Do

- Use Proper Naming
- Give Users & Admins Options
- Use Appropriate Workflow Behaviors
- Use Roles and Groups

Roles and Groups

Image: Workflow Conditions

The screenshot shows the 'Workflow Conditions' section of a Jira transition configuration. At the top, there are four tabs: 'Triggers 0', 'Conditions 2', 'Validators 0', and 'Post Functions 6'. The 'Conditions' tab is selected. Below the tabs, a dropdown menu is open, showing 'All of the following conditions'. Underneath, there are two listed conditions:

- Only users in group **jira-administrators** can execute this transition.
- Only users in project role **Administrators** can execute this transition.

Do's and Don'ts

Don't

- Add Unneeded Complexity
- Create Temporary or “Dead” Statuses
- Be Too Specific
- Create Illogical Statuses and Transitions

Illogical Statuses and Transitions

Image: Incorrect Expectation Example



Custom Workflow Planning

- Phased Approach
 - Example: Your company is signing a partnership agreement
 - *Open > In Review > In Execution > Closed*

Phased Approach

Open

In Review

In Execution

Closed

- Review Contract
- Research
- Communication
- Negotiation
- etc

- Collect Signatures
- File Records
- etc.

Interview

Chris, Owner, Business Strategy Company

Consulting Process:

“We start off with a search for candidates and identify ideal clients we’d like to work with. We acquire a lead through various lead generation methods. We do company research prior to meeting with the candidate. We book a meeting. At the meeting, we use our “Fact Finding Template” and “Performance Checklist.” After the meeting we develop a solution, plan, or resources for how the consultant can help the company. For ongoing contracts, we’ll have weekly, quarterly, semi-annual, and annual meetings for their quarterly and long-term goals.”

Custom Workflow Process

- Narrative
- Statuses
- Statuses + Forward Transitions
- Statuses + Forward + Alternate Transitions
- Transition Behaviors

Narrative

"We start off with a search for candidates and identify ideal clients we'd like to work with. We acquire a lead through various lead generation methods."

- Related statuses: Open

"We do company research prior to meeting with the candidate. We book a meeting. At the meeting, we use our "Fact Finding Template" and "Performance Checklist.""

- Related statuses: Planning

"After the meeting we develop a solution, plan, or resources for how the consultant can help the company."

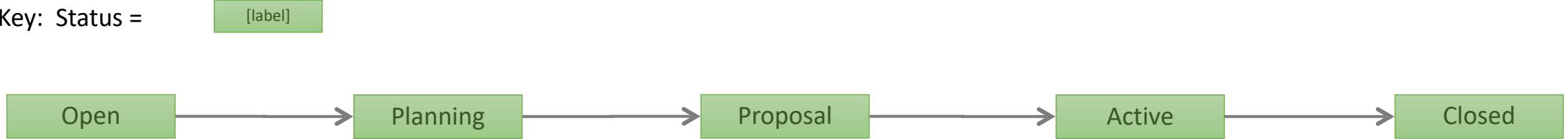
- Related statuses: Proposal

"For ongoing contracts, we'll have weekly, quarterly, semi-annual, and annual meetings for their quarterly and long-term goals."

- Related statuses: Active or Closed

Statuses

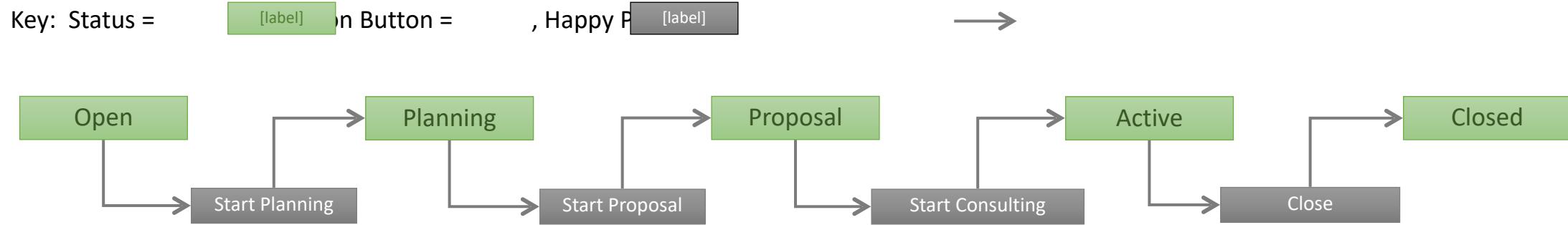
Key: Status =



In Words:

A lead is acquired and on creation is in the “Open” status. Research and meetings occur in the “Planning” status. Solutions are determined and pitched in the “Proposal” status. If the proposal is accepted, the issue moves to the “Active” status where all ongoing consulting occurs. When consulting is complete, the issues moves to the final “Closed” status.

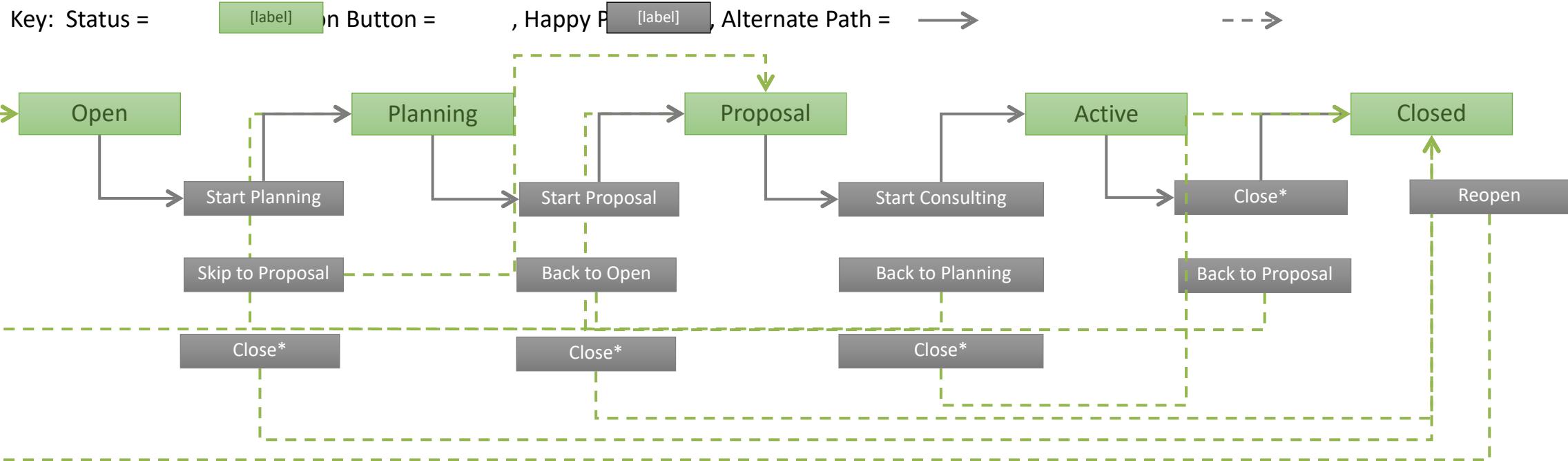
Statuses + Forward Transitions



In Words:

A lead is acquired and on creation is in the "Open" status. The user clicks the "Start Planning" transition button to move to the "Planning" status where all research and meetings occur. The user clicks the "Start Proposal" button to determine solutions and pitch them to the customer. If the proposal is accepted, the user clicks the "Start Consulting" button to move to the "Active" status where all ongoing consulting occurs. When consulting is complete, the user clicks the "Close" button to move to the final "Closed" status.

Statuses + Alternate Transitions



In Words:

A lead is acquired and on creation is in the “Open” status. The user clicks the “Start Planning” transition button to move to the “Planning” status. If no planning is needed, the user clicks “Skip to Proposal” to skip to the “Proposal” status. In the “Planning” status, the user clicks “Start Proposal” to move forward to the “Proposal” status or clicks “Back to Open” to move backwards to the “Open” status. In the “Proposal” status, the user clicks “Start Consulting” if the proposal is accepted. If the proposal is declined, the user clicks the “Close” button to move to the final “Closed” status. In the “Active” status, the user clicks the “Close” button when consulting is complete. In the “Closed” status, a user clicks the “Reopen” button to reopen the issue.

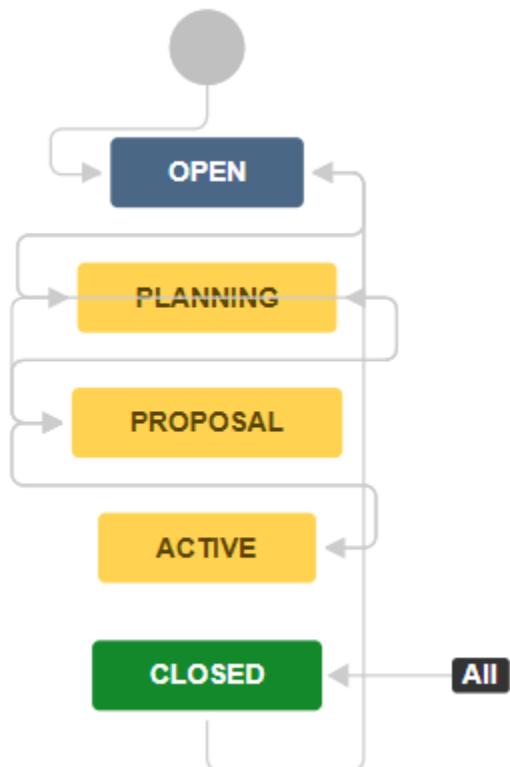
* The “Close” transition is global.

Transition Behaviors

| Status | Transition > Destination | Behaviors |
|--------|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| All | Close > Closed | <ul style="list-style-type: none">• Screen: Resolution• Post Function: Assign the issue to the reporter.• Post Function: Fire a Issue Closed event that can be processed by the listeners. |
| Closed | Reopen > Open | <ul style="list-style-type: none">• Post Function: The value of field Resolution will be cleared. |

Custom Workflow

Image: Diagram vs Text Mode



| Step Name (id) | Linked Status | Transitions (id) |
|----------------|---------------|-------------------------------------------------------------------------------------------------------|
| Open (1) | OPEN | Start Planning (11) >> PLANNING Skip to Proposal (21) >> PROPOSAL Close (91) >> Closed |
| Planning (2) | PLANNING | Start Proposal (31) >> PROPOSAL Back to Open (41) >> OPEN Close (91) >> Closed |
| Proposal (3) | PROPOSAL | Start Consulting (51) >> ACTIVE Back to Planning (61) >> PLANNING Close (91) >> Closed |
| Active (4) | ACTIVE | Back to Proposal (71) >> PROPOSAL Close (91) >> Closed |
| Closed (5) | CLOSED | Reopen (81) >> OPEN Close (91) >> Closed |

Workflow Concepts

Image: Diagram vs Text Mode

| Step Name (id) | Linked Status | Transitions (id) |
|-----------------|---------------|--------------------------------------------------------|
| To Do (1) | TO DO | Start Progress (11) >> IN PROGRESS |
| In Progress (2) | IN PROGRESS | Stop Progress (21) >> TO DO Done (31) >> DONE |
| Done (3) | DONE | Reopen (41) >> TO DO |



Draft vs Active

Image: Draft Mode

Software Simplified Workflow for Project DEMO **DRAFT** USED BY 1 PROJECT

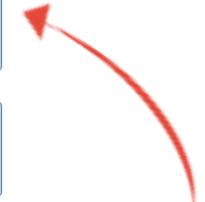
Generated by JIRA Software version 7.7.0-DAILY20180130085629. This workflow is managed internally by JIRA Software. Do not manually modify this workflow.

You are editing a draft. There are unpublished changes. **Publish** **Discard** [View Original](#)

This draft was last edited by **you** at 15/Feb/18 7:10 PM.

[Diagram](#) [Text](#) [Export ▾](#)

| Step Name (id) | Linked Status | Transitions (id) | Actions |
|----------------|---------------|-----------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| To Do (1) | TO DO | <i>To Do</i> (11) >> To Do <i>In Progress</i> (21) >> In Progress <i>Done</i> (31) >> Done | Add transition Edit View Properties |

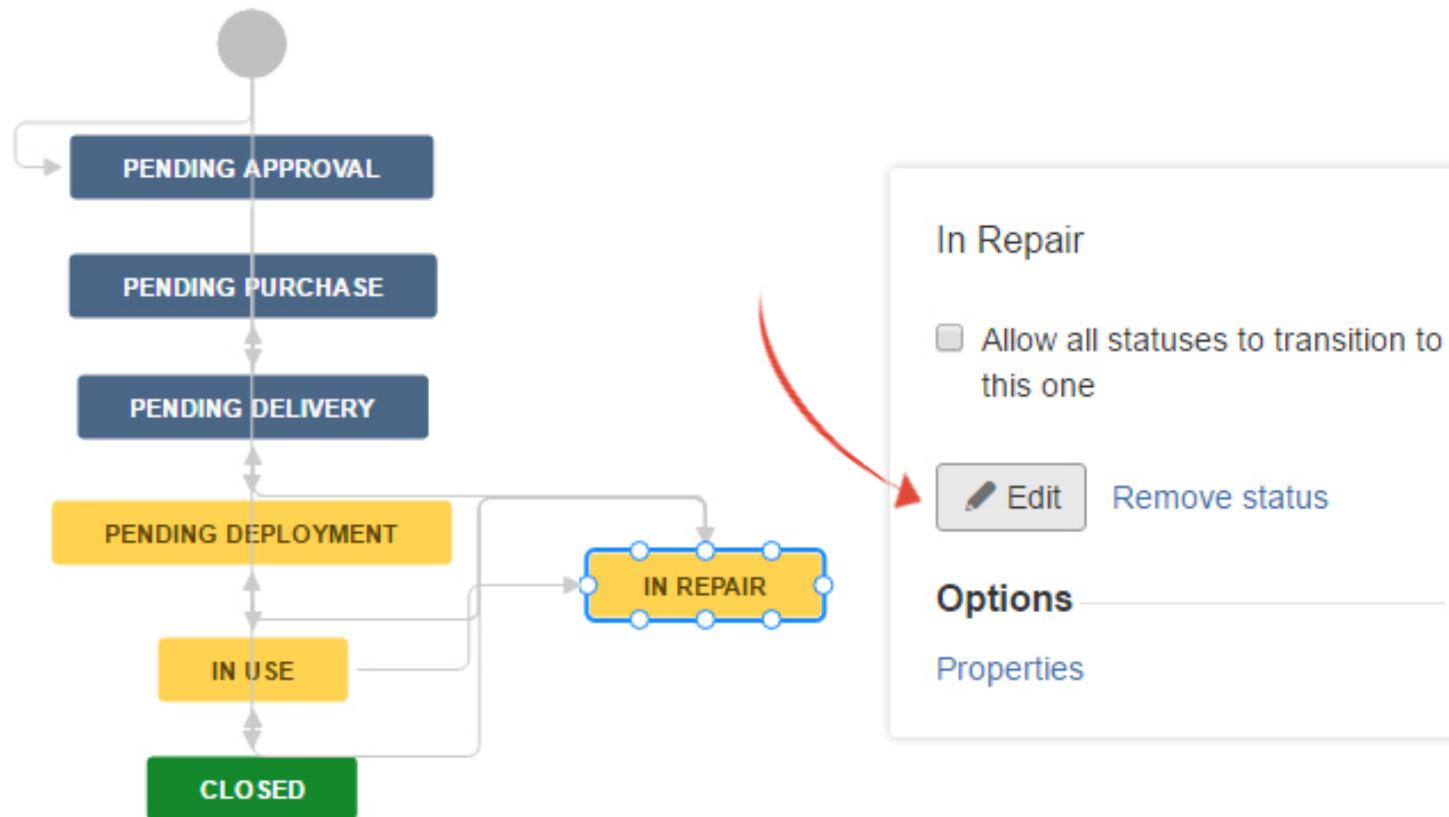


Workflow Editing Tips

- Make a Workflow Copy
- Build in a Test Environment
- Workflow Changes Impact JQL

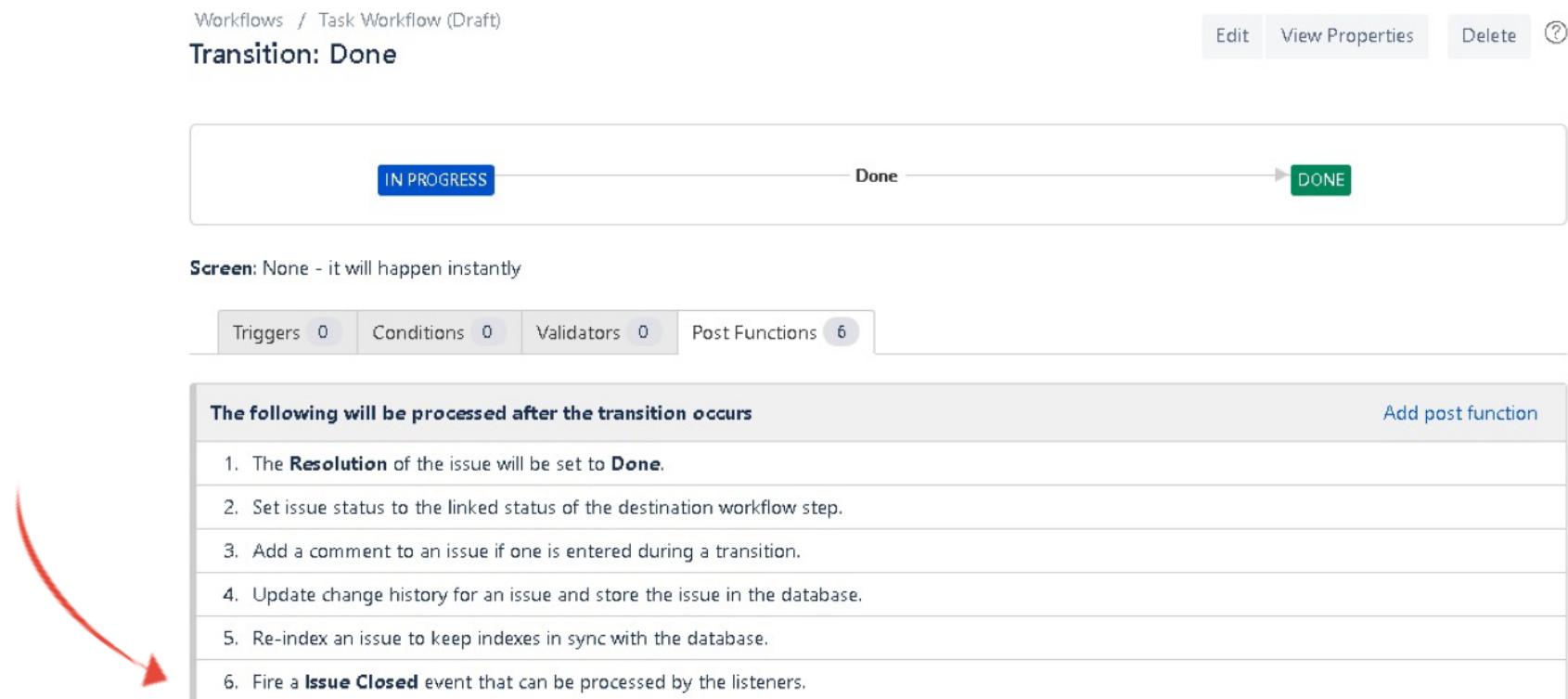
Workflow Assumptions

Image: Workflow Diagram Editor



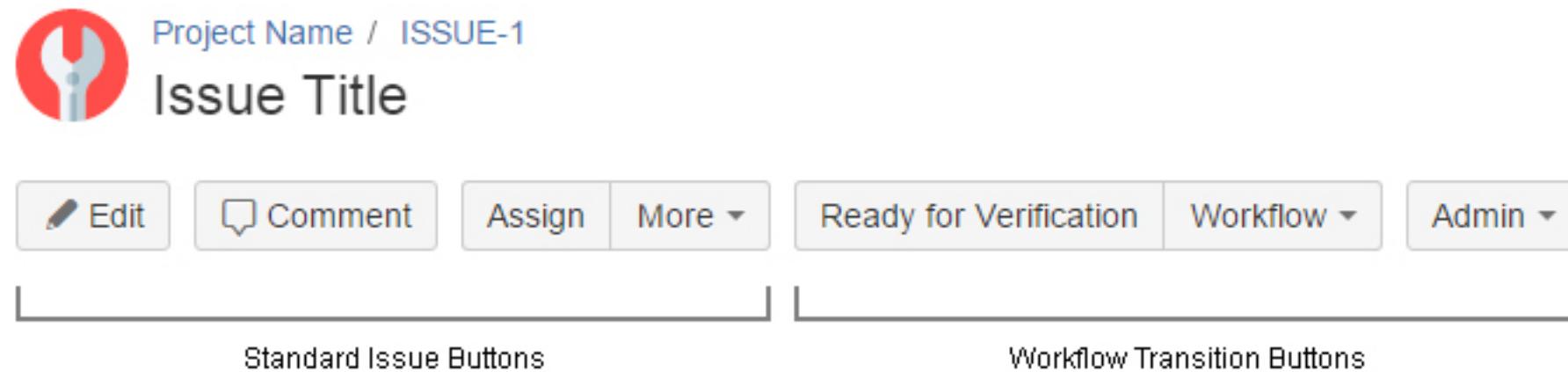
Workflow Assumptions

Image: Default Post Function



Transitions

Image: Standard Buttons vs Workflow Buttons



Transition Types

- Forward
- Backward
- Global
- Skip
- Administrative

Sample Workflow:

Open > In Review > In Execution > Closed

Transition Behaviors

Image: Triggers, Conditions, Validators, and Post Functions

The following will be processed after the transition occurs

1. The **Resolution** of the issue will be **cleared**.
2. Set issue status to the linked status of the destination workflow step.
3. Add a comment to an issue if one is entered during a transition.
4. Update change history for an issue and store the issue in the database.
5. Re-index an issue to keep indexes in sync with the database.
6. Fire a **Work Started On Issue** event that can be processed by the listeners.

Default Workflows

Image: Software – Scrum Workflow

Scrum software development

Use this project to manage your Agile development work. Create a backlog, organise work into sprints, check progress using reports, and more. This project includes a Scrum board, a basic Agile workflow and issue type configuration, which you can change later on.

| ISSUE TYPES | WORKFLOW |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------|
| <ul style="list-style-type: none"><input type="checkbox"/> Bug<input checked="" type="checkbox"/> Task<input type="checkbox"/> Sub-task<input type="checkbox"/> Story<input type="checkbox"/> Epic | <pre>graph LR; A[TO DO] --> B[IN PROGRESS]; B --> C[DONE]</pre> |

[Back](#) [Select](#) [Cancel](#)

Default Workflows

Image: Software – Kanban Workflow

Kanban software development

Use this project to optimise the flow of work in your development project. Add constraints to work-in-progress, analyze how long it takes to complete issues, find bottlenecks in your process, and more. This project includes a Kanban board, a basic Agile workflow and issue type configuration, which you can change later on.

| ISSUE TYPES | WORKFLOW |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"><input checked="" type="checkbox"/> Bug<input checked="" type="checkbox"/> Task<input checked="" type="checkbox"/> Sub-task<input checked="" type="checkbox"/> Story<input checked="" type="checkbox"/> Epic | <pre>graph LR; BACKLOG[BACKLOG] --> SELECTED[SELECTED FOR DEV...]; SELECTED --> IN_PROGRESS[IN PROGRESS]; IN_PROGRESS --> DONE[DONE]</pre> |

Back **Select** **Cancel**

Default Workflows

Image: Software – Basic Workflow

Basic software development

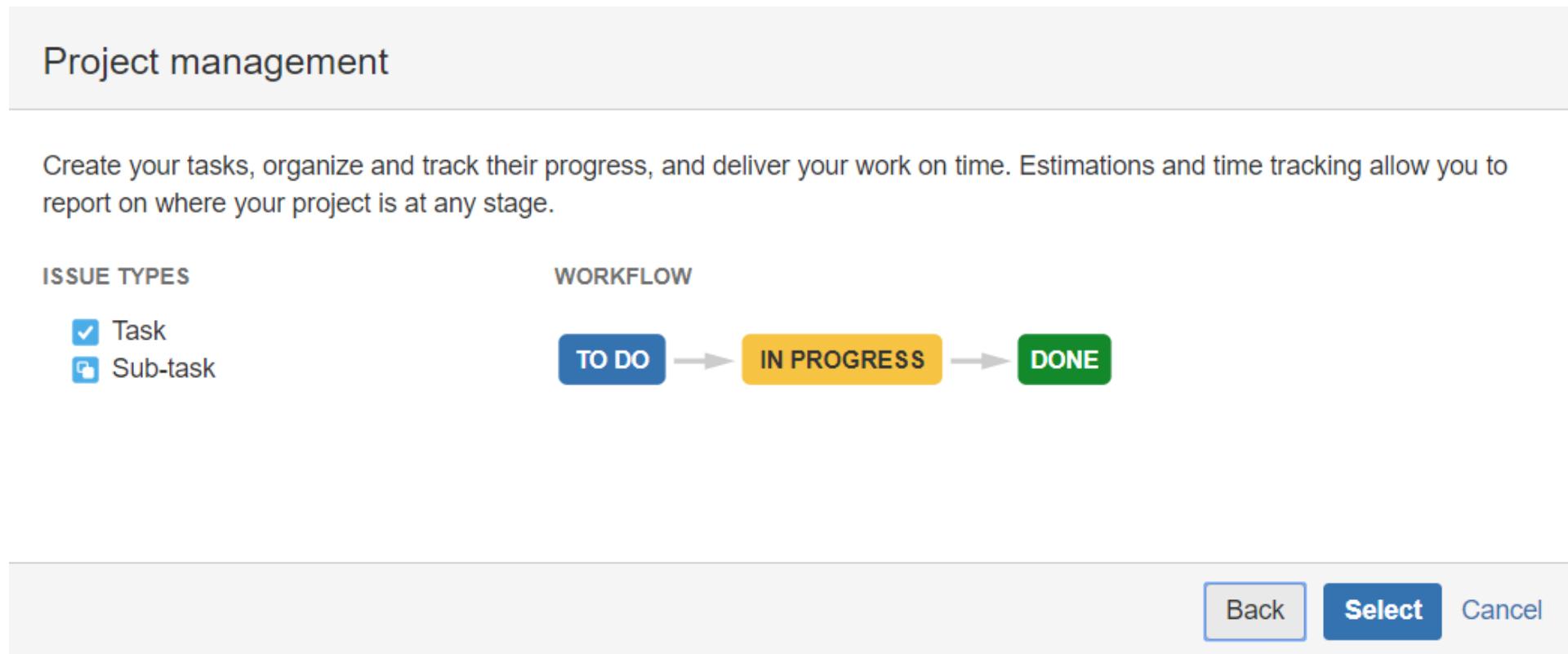
Use this project to work on new features for your product and also track any bugs. This project provides you with a basic workflow and issue type configuration, which you can change later on.

| ISSUE TYPES | WORKFLOW |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"><input checked="" type="checkbox"/> Bug<input checked="" type="checkbox"/> Task<input checked="" type="checkbox"/> Sub-task<input checked="" type="checkbox"/> Improvement<input checked="" type="checkbox"/> New Feature<input checked="" type="checkbox"/> Epic | <pre>graph LR; A[TO DO] --> B[IN PROGRESS]; B --> C[IN REVIEW]; C --> D[DONE]</pre> |

Back **Select** **Cancel**

Default Workflows

Image: Business – Project Management Workflow



Default Workflows

Image: Business – Task Workflow

Task management

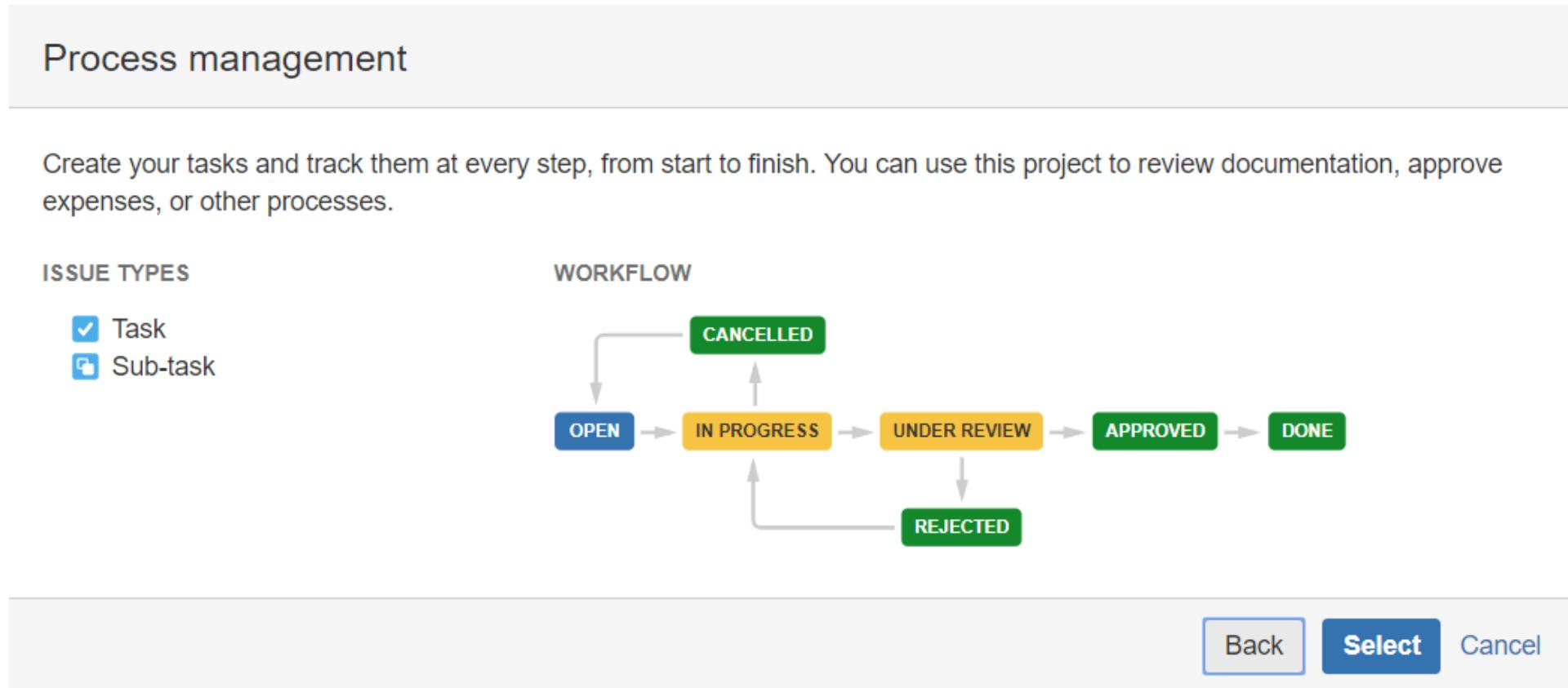
Create simple tasks, organize them and get them done. You can use this project to manage your tasks or assign them to someone else.

| ISSUE TYPES | WORKFLOW |
|-------------------------------------------------------------------------------|----------------------------------------|
| <input checked="" type="checkbox"/> Task <input type="checkbox"/> Sub-task | TO DO → DONE |

Back Select Cancel

Default Workflows

Image: Business – Process Management Workflow



Multiple Workflows

Image: Issue Type to Workflow Matrix

| Issue Type | Description | Workflow |
|------------------------|---------------------------------|------------|
| \$ Contract | Third party agreements | ↳ Contract |
| ✓ Task | A task that needs to be done | ↳ Task |
| ✓ Terms and Conditions | Terms for products and services | ↳ Terms |
| ⌚ Subtask SUB-TASK | | ↳ Task |