Foundations of Software Engineering

Lecture 6: Requirements Solicitation and Documentation

Christian Kästner

institute for SOFTWARE RESEARCH

Learning goals

- Basic proficiency in executing effective requirements interviews
- Understand tradeoffs of different documentation strategies
- Document requirements using use cases
- Recognize and resolve conflicts with priorities

REQUIREMENTS ELICITATION

Case: Web video

“We want to sell videos on the web.”

Typical Steps

- Identify stakeholders
- Understand the domain
  - Analyze artifacts, interact with stakeholders
- Discover the real needs
  - Interview stakeholders
- Explore alternatives to address needs

Question

- Who is the system *for*?
- Stakeholders:
  - End users
  - System administrators
  - Engineers maintaining the system
  - Business managers
  - …who else?

Stakeholder

- Any person or group who will be affected by the system, directly or indirectly.
- Stakeholders may disagree.
- Requirements process should trigger negotiation to resolve conflicts.
  - (We will return to conflicts).

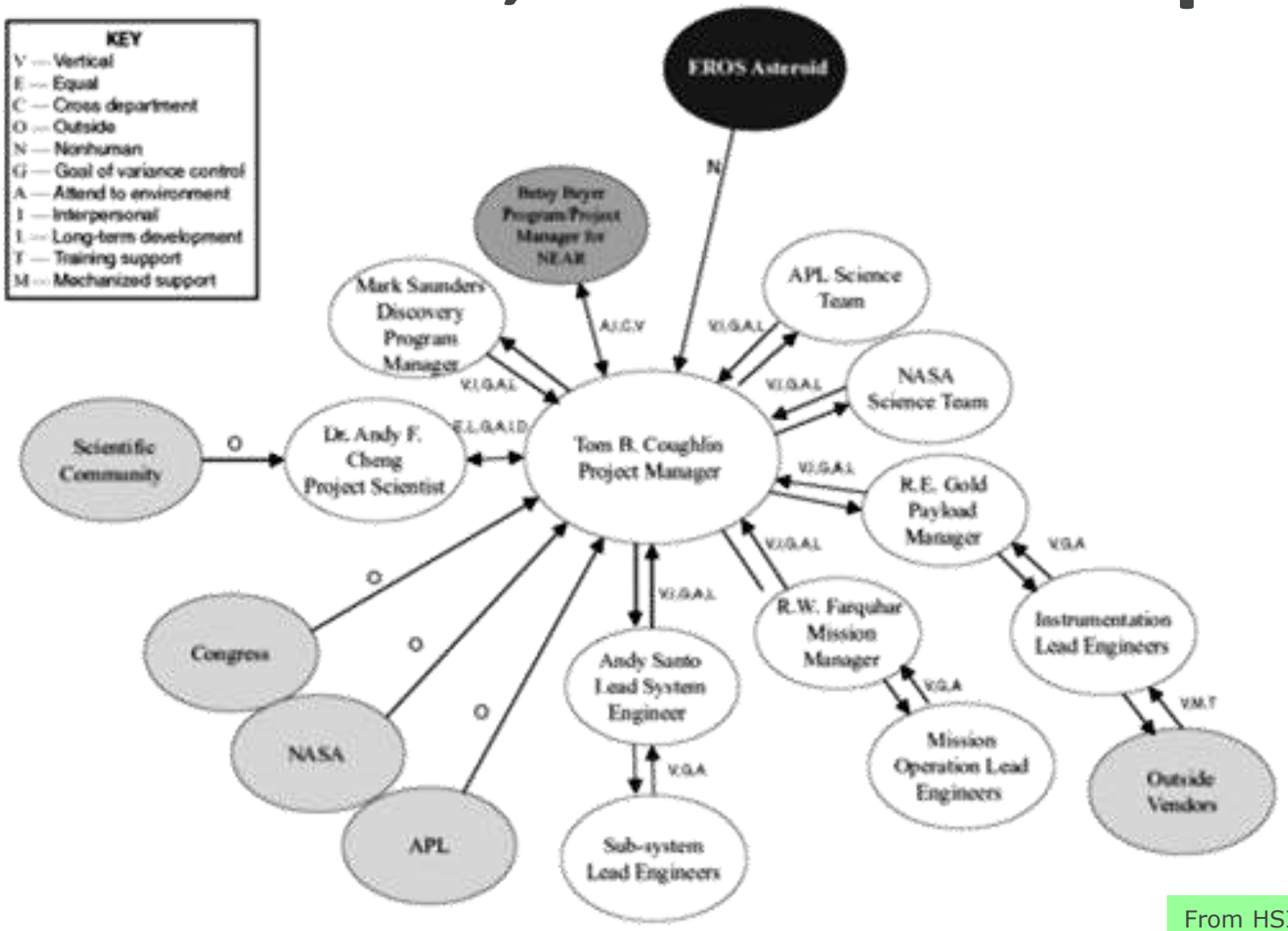
Defining actors/agents

- An actor is an entity that interacts with the system for the purpose of completing an event [Jacobson, 1992].
  - Not as broad as stakeholders.
- Actors can be a user, an organization, a device, or an external system.

Stakeholder analysis: criteria for identifying relevant stakeholders

- Relevant positions in the organization
- Effective role in making decisions about the system
- Level of domain expertise
- Exposure to perceived problems
- Influence in system acceptance
- Personal objectives and conflicts of interest

Stakeholders, a NASA example



From HSI NAP 11893

FIGURE 6-3 Role network for National Aeronautics and Space Administration (NASA's) Near Earth Asteroid Rendezvous project.

Challenges

- Distributed knowledge
- Conflicting knowledge
- Difficult to access sources
- Communication barriers (cultural, terminology, backgrounds)
- Hidden needs, tactic knowledge
- Politics, unstable conditions

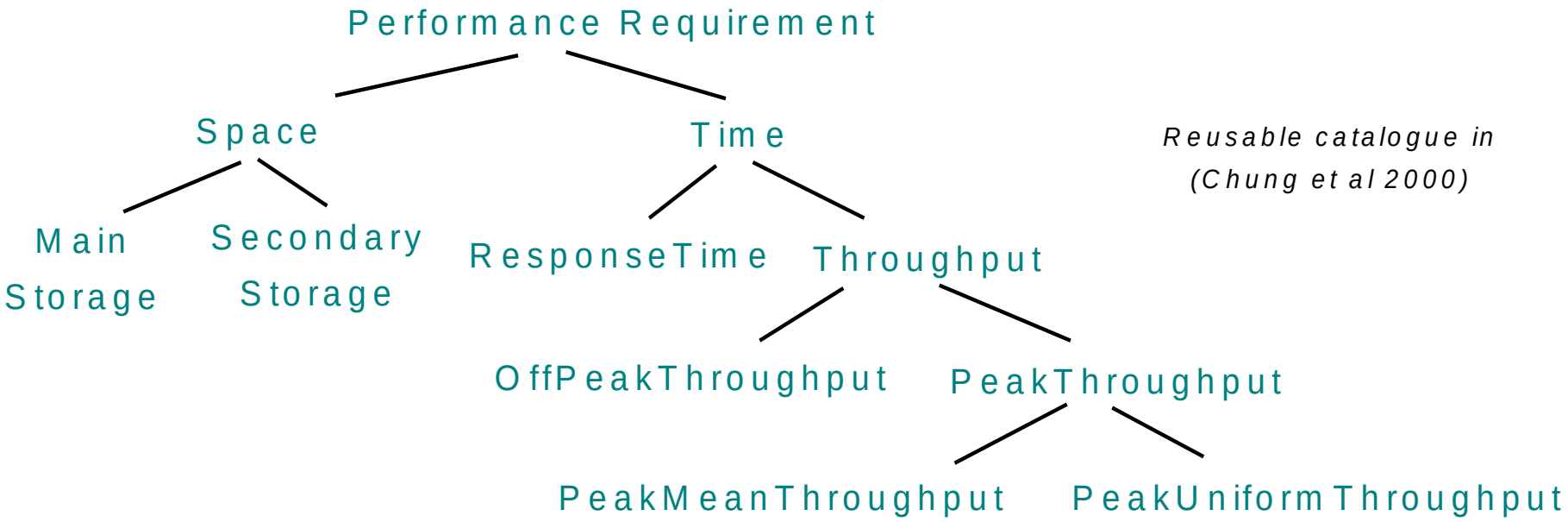
Examples? Mitigation strategies?

Studying Artifacts (Content Analysis)

- Learn about the domain
  - Books, articles, wikipedia
- Learn about the system to be replaced
  - How does it work? What are the problems? Manuals? Bug reports?
- Learn about the organization
- Knowledge reuse from other systems?

Checklists (Domain-independent knowledge)

- Consider list of qualities for relevance, e.g. privacy, security, reliability, …



Reusable catalogue in (Chung et al 2000)

Interviews

Interview Process

- Identify stakeholder of interest and target information to be gathered
- Conduct interview
  - (structured/unstructured, individual/group)
- Record + transcribe interview
- Report important finding
- Check validity of report with interviewee

Interview Advice

- Get basic facts about the interviewee before (role, responsibilities, …)
- Review interview questions before interview
- Begin concretely with specific questions, proposals; work through prototype or scenario
- Be open-minded; explore additional issues that arise naturally, but stay focused on the system.
- Contrast with current system/alternatives. Explore conflicts and priorities
- Plan for follow-up questions

Example: Identifying Problems

- What problems do you run into in your day-to-day work? Is there a standard way of solving it, or do you have a workaround?
  - Why is this a problem? How do you solve the problem today? How would you ideally like to solve the problem?
- Keep asking follow-up questions ("What else is a problem for you?", "Are there other things that give you trouble?") for as long as the interviewee has more problems to describe.

- So, as I understand it, you are experiencing the following problems/needs (describe the interviewee's problems and needs in your own words – often you will discover that you do not share the same image. It is very very common to not understand each other even if at first you think you do).
- Just to confirm, have I correctly understood the problems you have with the current solution?
- Are there any other problems you're experiencing? If so, what are they?

http://reqtest.com/requirements-blog/how-to-use-interviews-to-gather-requirements/

Example Questions: The User Environment

- Who will be the users of the system?
- What level of education or training do the users have?
- What computer skills do the users have?
- Are users familiar with this type of IT system?
- What technical platforms do they use today?
- Do you know of any plans for future systems or platforms?
- What other IT systems does the organization use today that the new system will need to link to?
- What are your expectations regarding system usability?
- What training needs do you expect for the future system?
- What kind of documentation do you expect?

http://reqtest.com/requirements-blog/how-to-use-interviews-to-gather-requirements/

Interview Tradeoffs

- Strengths
  - What stakeholders do, feel, prefer
  - How they interact with the system
  - Challenges with current systems
- Weaknesses
  - Subjective, inconcistencies
  - Capturing domain knowledge
  - Familiarity
  - Technical subtlety
  - Organizational issues, such as politics
  - Hinges on interviewer skill

Capturing v. Synthesizing

- Engineers acquire requirements from many sources
  - Elicit from stakeholders
  - Extract from policies or other documentation
  - Synthesize from above + estimation and invention
- Because stakeholders do not always know what they want, engineers must…
  - Be faithful to stakeholder needs and expectations
  - Anticipate additional needs and risks
  - Validate that “additional needs” are necessary or desired

Guidelines for effective interviews

- Identify the right interviewee sample for full coverage of issues
  - different responsibilities, expertise, tasks, exposure to problems
- Come prepared, to focus on right issue at right time
  - backgound study first
  - predesign a sequence of questions for this interviewee
- Centre the interview on the interviewee's work & concerns
- Keep control over the interview
- Make the interviewee feel comfortable
  - Start: break ice, provide motivation, ask easy questions
  - Consider the person too, not only the role
  - Do always appear as a trustworthy partner

Guidelines for effective interviews

- Be focused, keep open-ended questions for the end
- Be open-minded, flexible in case of unexpected answers
- Ask why-questions without being offending
- Avoid certain types of questions ...
  - opiniated or biased
  - affirmative
  - obvious or impossible answer for this interviewee
- Edit & structure interview transcripts while still fresh in mind
  - including personal reactions, attitudes, etc
- Keep interviewee in the loop
  - co-review interview transcript for validation & refinement

Ethnography

Observation & ethnography

- Observe people using the current system
- Passive: no interference with task performers
  - Watch from outside, record (notes, video), edit transcripts, interpret
  - Protocol analysis: task performers concurrently explain
- Active observation: you get involved in the task, even become a team member
- Ethnographic studies: over long periods of time, try to discover emergent properties of social group involved

Observation & ethnography - Tradeoffs

- May reveal ...
  - tacit knowledge that would not emerge otherwise
  - hidden problems through tricky ways of doing things
  - culture-specific aspects to be taken into account
- Contextualization of acquired info

- Slow & expensive: to be done over long periods of time, at different times, under different workload conditions
- Potentially inaccurate (people behave differently when observed)
- Data mining problem, interpretation problem
- Focus on system-as-is

Group sessions

- More perception, judgement, invention from interactions within group of diverse people
- Elicitation takes place in series of group workshops (a few days each) + follow-up actions
  - audiovisuals, wall charts to foster discussion, record outcome
- Structured group sessions:
  - Each participant has a clearly defined role (leader, moderator, manager, user, developer, ...)
  - Contributes to req elaboration according to his/her role, towards reaching synergies
  - Generally focused on high-level reqs
  - Variants: focus groups, JAD, QFD, ...
- Unstructured group sessions (brainstorming):

Group sessions - Tradeoffs

- Less formal interactions than interviews
  - => may reveal hidden aspects of the system (as-is or to-be)
- Potentially ...
  - wider exploration of issues & ideas
  - more inventive ways of addressing problems
- Synergies => agreed conflict resolutions

- Group composition is critical ...
  - time consuming for key, busy people
  - heavily relying on leader expertise & skills
  - group dynamics, dominant persons => biases, inadequacies
- Risk of ...
  - missing focus & structure => rambling discussions, little concrete outcome, waste of time
  - superficial coverage of more technical issues

Combining techniques

- Many combined and more specific approaches
- For example Contextual Inquiry:
  - workplace observation +
  - open-ended interviews +
  - prototyping

- Yai: non-profit; most employees in social work field
- Currently sells training DVDs for companies on issues related to individuals with developmental disabilities.
- **“Do you know how we can sell course materials online?”**

RESOLVING CONFLICTS

Types of inconsistency

- Terminology clash: same concept named differently in different statements
  - e.g. library management: “borrower” vs. “patron”
- Designation clash: same name for different concepts in different statements
  - e.g. “user” for “library user” vs. “library software user”
- Structure clash: same concept structured differently in different statements
  - e.g. “latest return date” as time point (e.g. Fri 5pm) vs. time interval (e.g. Friday)

Types of inconsistency, 2

- Strong conflict: statements not satisfiable together
  - e.g. “participant constraints may not be disclosed to anyone else” vs. “the meeting initiator should know participant constraints”
- Weak conflict (divergence): statements not satisfiable together under some boundary condition
  - “patrons shall return borrowed copies within X weeks” vs “patrons shall keep borrowed copies as long as needed” contradict only if “needed>x weeks”

Handling inconsistencies

- Terminology, designation, structure: Build glossary
- Weak, strong conflicts: Negotiation required
  - Cause: different objectives of stakeholders => resolve outside of requirements
  - Cause: quality tradeoffs => explore preferences

Examples?

Resolution Strategies

- Various specific processes, heuristics, and techniques exist for identifying and resolving conflicts. See literature for details.

Requirements Traceability

- Keep connections between requirements
- What follows from what

Requirements prioritization

- Cost, time, and other limits
- Dependencies among requirements
- Nice to have

- Strategies to base on value contribution

PROTOTYPES, MOCKUPS, STORIES

High- vs low- fidelity mockups

Why prototypes/mockups?
How to use?

Mockups, Prototypes, Stories

- Humans: better at recognizing whether a solution is correct than solving the problem from a blank page.
- **Mock-ups/prototypes** help explore uncertainty in the requirements.
  - Validate that we have the right requirements.
  - Elicit requirements at the “borders” of the system.
  - Assert feasibility of solution space.
  - Get feedback on a candidate solution.
- “I’ll know it when I see it”

Rapid prototyping

- Throw-away: developed to learn more about a problem, not intended for actual use.

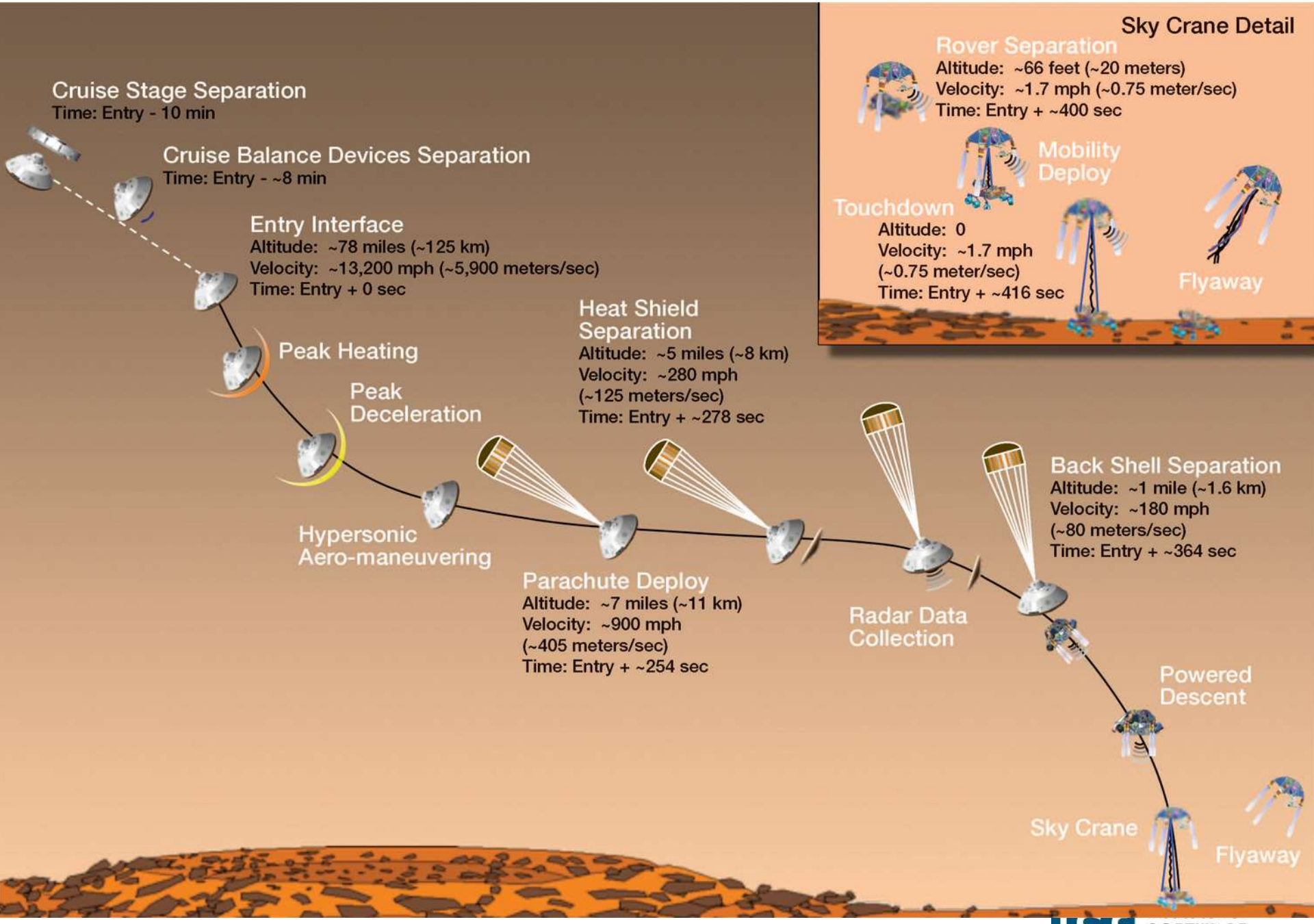
- Evolutionary: intended to be incorporated into the final product.

Storyboarding and scenarios

Story

- **Who** the players are
- **What** happens to them
- **How** it happens through specific episode
- **Why** this happens
- **What** if such and such an event occurs
- **What** could go wrong as a consequence

- **Storyboards illustrate scenarios**: a typical sequence of interaction among system components that meets an implicit objective.
  - **Storyboards** explicitly cover at least **who**, **what**, and **how**.
- Different types:
  - Positive vs negative (should and should not happen)
  - Normal vs abnormal
- As part of elicitation:
  - Learn about current or proposed system by walking through real-life or hypothetical sequences
  - Can ask specific questions
  - Elicit the underlying objectives, generalize into models of desired behaviors.
  - Identify and resolve conflicts
- Pluses: Concrete, support narrative description
- Minuses: inherently partial.



Scenarios

Test cases

- Questions to consider
  - What tasks does the actor perform?
  - What information is accessed and modified, and where does it come from?
  - What are obligations on the actor to inform the system?
  - What are obligations of the system to inform the actor?
- Heuristics
  - Vertical – one worked-out specific scenario, to understand how to engage the user/stakeholder
  - Horizontal – multiple, less-detailed scenarios, to assess scope and context
  - Mock-ups
  - Alternatives
  - Can be **passive** or **active**.

DOCUMENTING REQUIREMENTS

Many different forms

- Informal vs formal
- Unstructured vs structured
- Text vs diagrams

- Structured text common in practice
- Tool supported for traceability and process integration

Software Requirements Specification (SRS)

- Formal requirements document
- Several standards exists
- Often basis for contracts

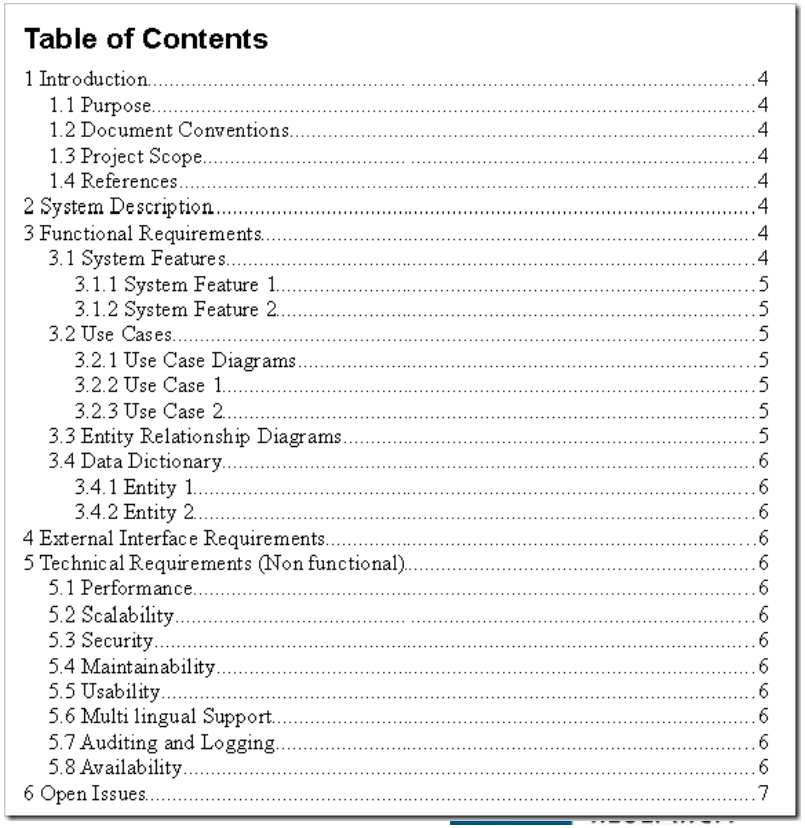
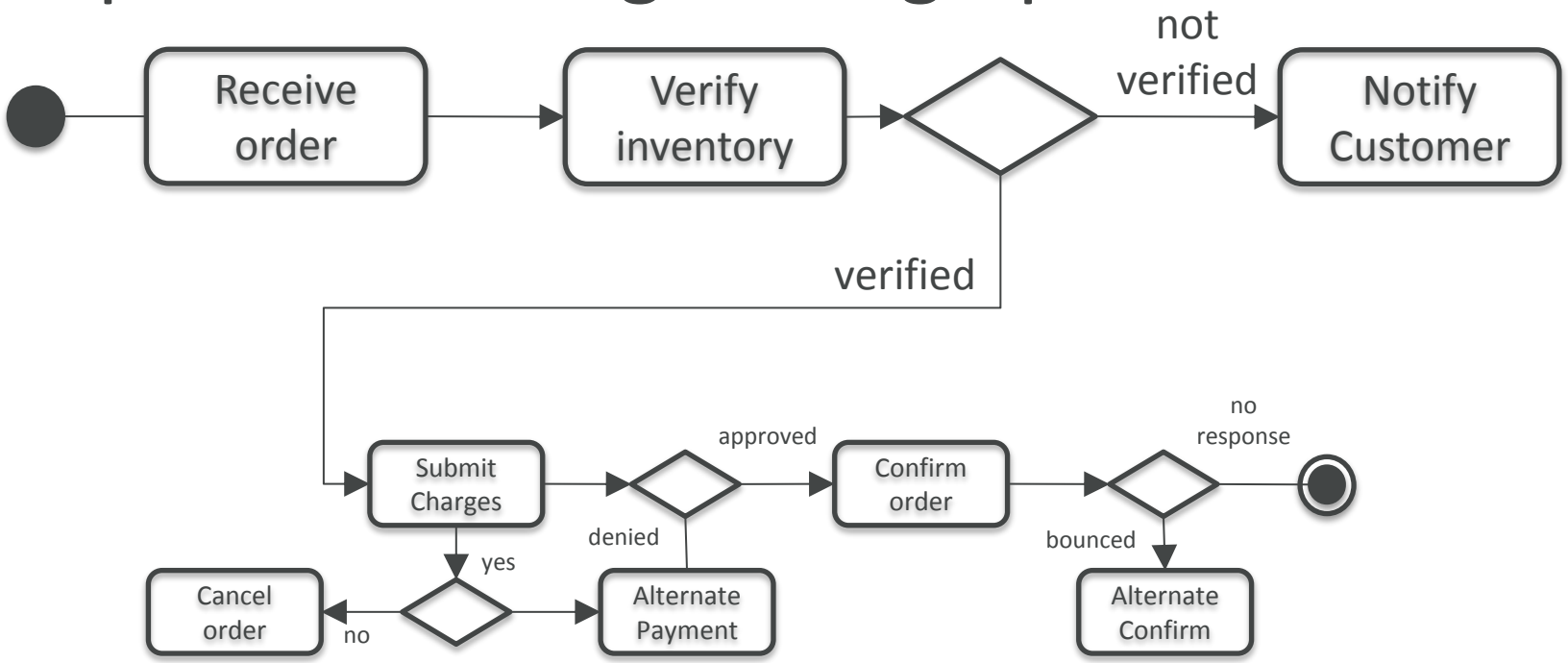


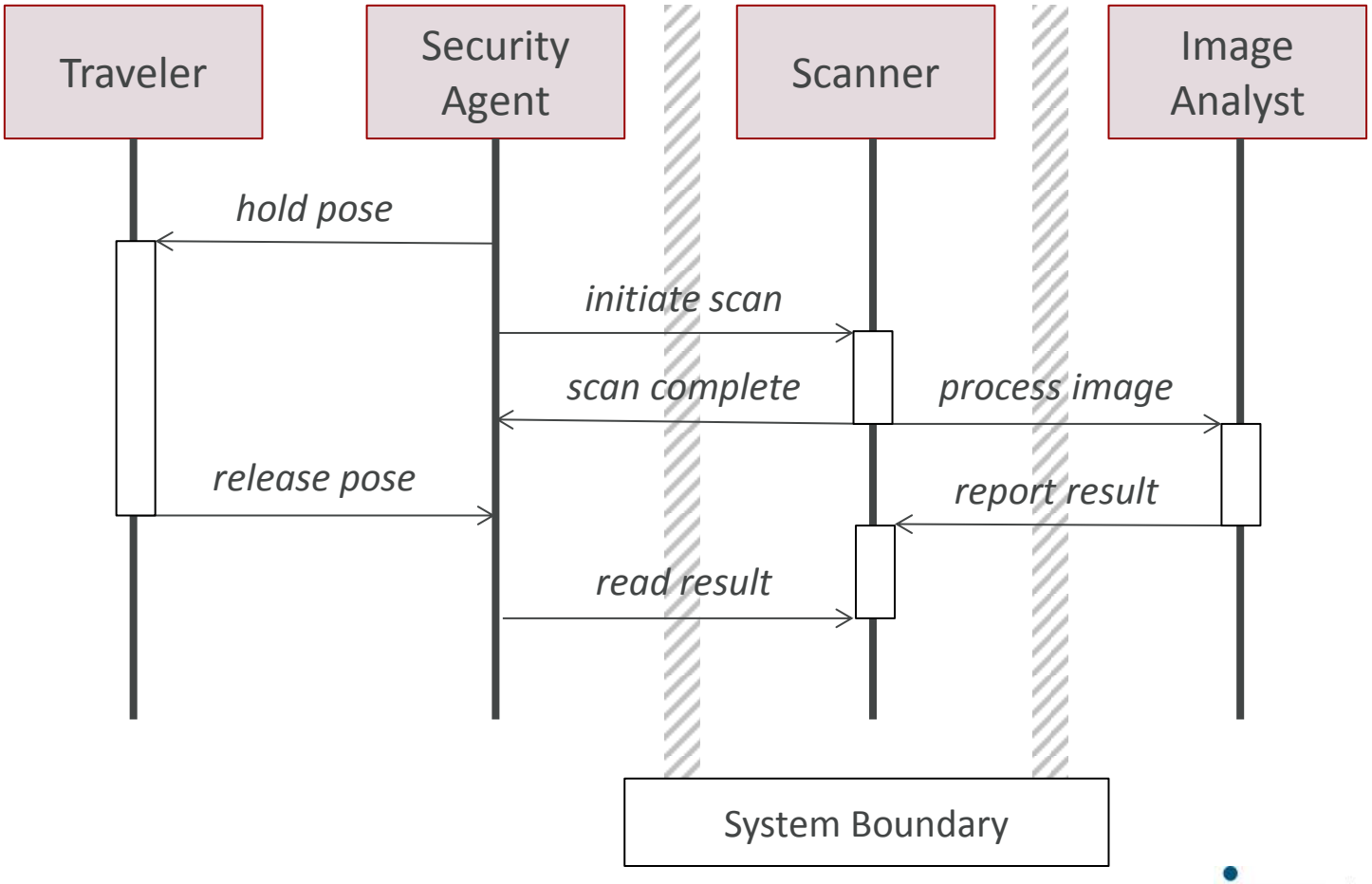
Table of Contents

Activity Diagrams

Activity diagrams (or flow charts) represent the logic in a graph notation



Sequence Diagramming



Storyboarding and scenarios

Formal specifications

- Logical expressions of shared actions at the interface of the machine
- Includes linking domain properties and agent actions as pre- and post-conditions

$$\forall\, s\, \forall\, c(\text{enrolled}(s, c) \Rightarrow \text{student}(s) \wedge \text{course}(c))$$

Grounding formal specifications

- **Able**: Two important basic types are *student* and *course*. There is also a binary relation *enrolled*.
- Able defines these elements as follows:

$$\forall\, s\, \forall\, c(\text{enrolled}(s, c) \Rightarrow \text{student}(s) \wedge \text{course}(c))$$

- **Baker**: Do only students enroll in courses? I don't think that's true.
- **Able**: But that's what I mean by student!

Designations as explanations

- If person is enrolled in a course, then the person is a student:

$$\forall\, s\, \forall\, c(\text{enrolled}(s, c) \Rightarrow \text{student}(s) \wedge \text{course}(c))$$

- A person is a student, if and only if, there is a course where the student is enrolled

$$\forall\, s\, (\text{student}(s) \Leftrightarrow \exists\, c\ \text{enrolled}(s, c))$$

Use case

- *Text story* of an actor using a system to meet goals.
- *Use cases are not diagrams, they are text.*
- Primarily serve as functional requirements (by contrast/in conjunction with “the system shall” statements.)

Use Case Name	(Title)
Scope	System under design
Level	User level, subprocess level
Primary actor	(actors can be primary, supporting, or offstage)
Stakeholders, interests	Important! A use case should include everything necessary to satisfy the stakeholders' interests.
Preconditions	What must always be true before a scenario begins. Not tested; assumed. Don't fill with pointless noise.
Success guarantees.	Aka post conditions
Main success scenario	Basic flow, "happy path", typical flow. Defer all conditions to the extensions. Records steps: interaction between actors, a validation, a state change by the system.
Extensions	Aka alternate flows. Usually the majority of the text. Sometimes branches off into another use case.
Special requirements	Where the non-functional/quality requirements live.
Technology and data variations list	Unavoidable technology constraints; try to keep to I/O technologies.
Frequency of occurrence	
Miscellaneous	

Use cases

- We talk about many types, at different granularities:
  - Full use case model (whole-system, higher-level)
  - “Agile” use case: small, concrete pieces of system functionality to be implemented (sometimes conflated with “user stories”)
- Used at multiple stages:
  - Requirements elicitation (illustrated, validate, requirements; highlight conflicts, prioritize requirements, etc).
  - Requirements documentation.
  - Concrete design: UML diagrams.

Industrial Requirements Tools

'Stakeholder Requirements' current 2.2 (Review Phase 3) in /New Family Car Project/Requirements (Formal module) - DO...

File Edit View Insert Link Analysis Table Tools Discussions User RQM Help

View: 01 - Collect reqts | All levels

- Stakeholder Requirements
  - 1 Introduction
  - 2 User types
    - 2.1 Nationalities
      - The car will be usec
    - 2.2 User sizes
      - People come in all s
  - 3 Requirements
    - 3.1 Capability Requirem
      - 3.1.1 Carrying Capa
      - 3.1.2 Cost Points
      - 3.1.3 Movement
      - 3.1.4 Fuel economy
      - 3.1.5 Safety
      - 3.1.6 Noise levels
      - 3.1.7 Ease of Acces
      - 3.1.8 Visibility
      - 3.1.9 Equipment ma
      - 3.1.10 Entertainmer
      - 3.1.11 Maintenance
      - 3.1.12 Servicing
      - 3.1.13 Indication rec
      - 3.1.14 Terrain
      - 3.1.15 Refueling
    - 3.2 Constraint Requirem
      - 3.2.1 Availability
      - 3.2.2 Lifetime
      - 3.2.3 Security
      - 3.2.4 Accessories

Reqt	Car user requirements	Priority	Acceptability	Queries
TRN-CSR-83	Users shall be able to receive a warning when a service is due.	Mandatory	Acceptable	(JC) What color indicators are we using for the warning system? Are we going to put requirements in place to accomidate this system?
TRN-CSR-84	**3.1.13 Indication requirements**	N/A		
TRN-CSR-85	The user shall be able to see at all times an indication of speed to within + or - 1%.	1	Acceptable	
TRN-CSR-86	The user shall be able to see at all times an indication of engine revolutions to within + or - 1%.	2	Acceptable	
TRN-CSR-92	The user shall be able to obtain direction to go information. 	2	Acceptable	

Username: lbus | Exclusive edit mode

Summary

- Many solicitation strategies, including document analysis, interviews, and ethnography
- Do not underestimate the challenge of interviews
- Resolving conflicts
- Using prototypes to enhance discussions and decision making
- Many documentation strategies; our focus *use cases*

Further Reading

- Larman, Craig. *Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design and Interative Development*. Pearson, 2012. Chap. 6
- Van Lamsweerde A. Requirements engineering: From system goals to UML models to software. John Wiley & Sons; 2009. Chapter 2-4