A: Getting Started

The Fuzzy Front End

Sometimes a development effort starts like this:

You lot start coding! I'll go and find out what they want.

The Fuzzy Front End

And sometimes it starts with a formal, complete specification containing

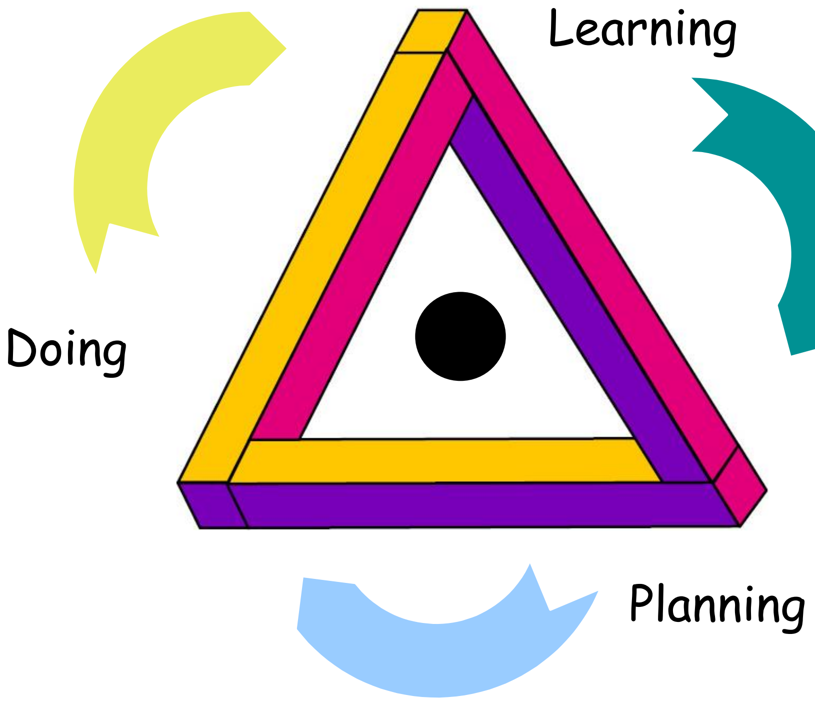
- charter,
- context, and
- theory of operation.

Every requirement is:

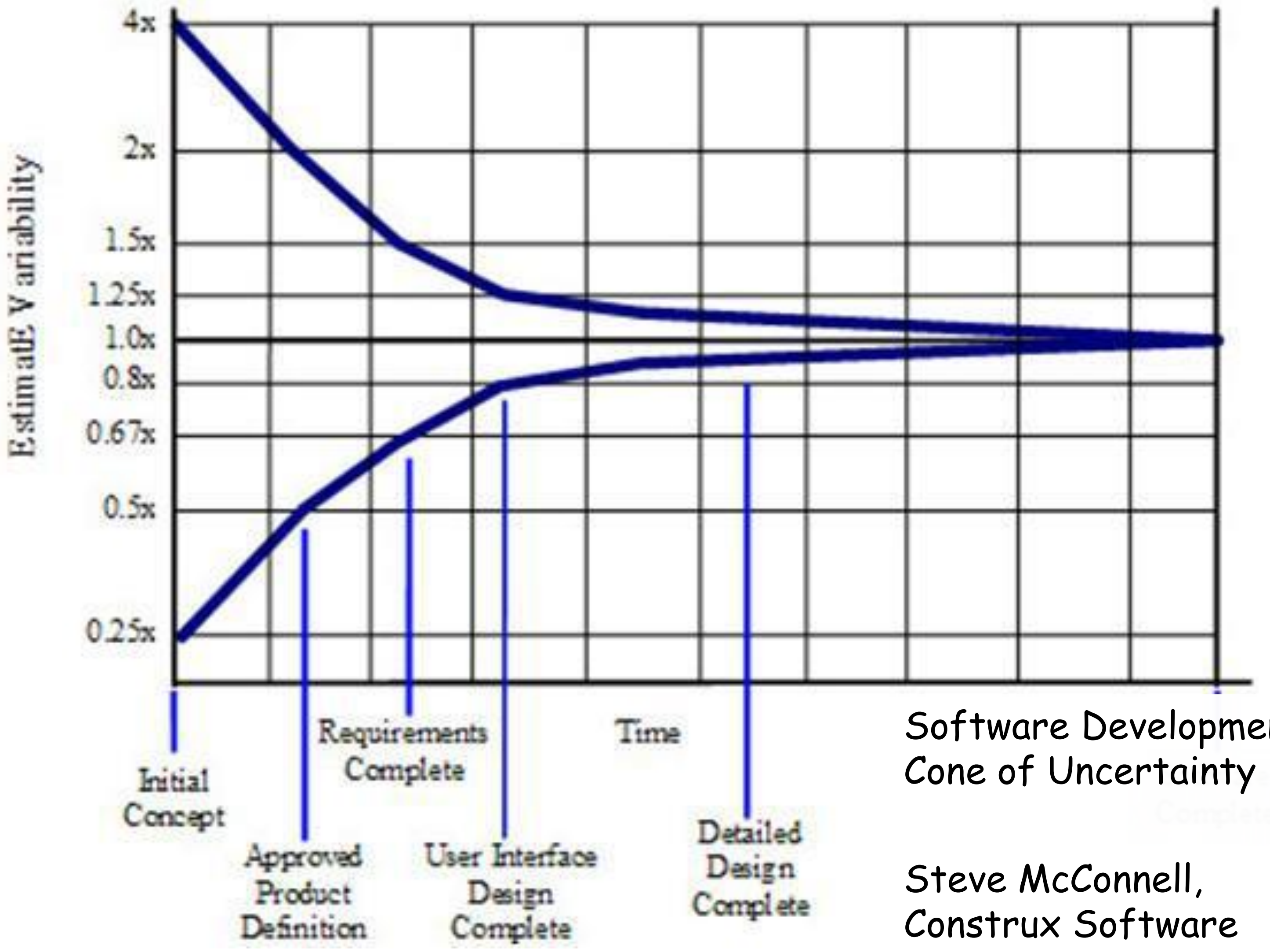
- Identified
- Unique
- Coherent
- Unambiguous
- Testable

Abstraction

To build models, you have to be able to gather enough information to make good abstractions.



Cone of Uncertainty



Software Development's
Cone of Uncertainty

Steve McConnell,
Construx Software

Levels of Commitment

Consequently, we must commit incrementally.

- Natural language and informal diagrams
  - Use cases
  - Activity diagrams
  - Sequence diagrams

  } This course

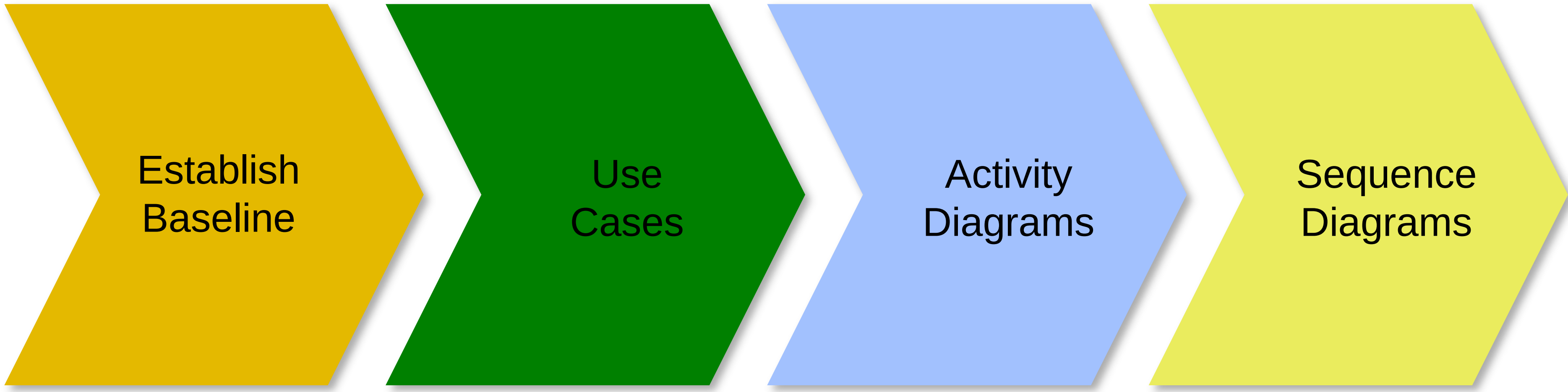
- Structural models
  - Components & Interfaces
  - Class models
  - Data types
- Behavioral models
  - State models
  - Activities

Requirements Clarification Process

The process is:

- Find all your people, resources, practices, etc.
- Find out what the system-as-a-whole does
- Determine the precise behavior of each use case
- And establish how it communicates with others

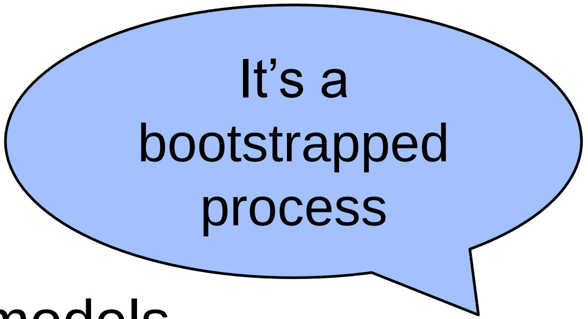
But it's really all about learning about the problem.



Requirements Clarification Process

We'll show you how to:

- determine what you have, and
- how well it meets your needs
- gather information to build executable models
- investigate questionable use cases
- organize information ready to build executable models



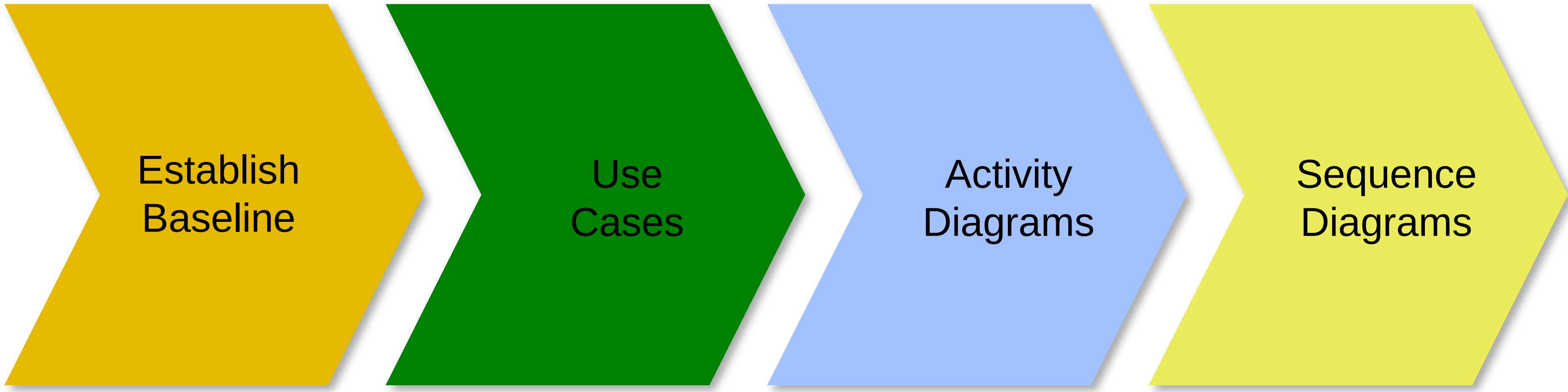


Table of Contents

1. Kick Off

1

Kick Off

To start the project, we need to know:

- Charter
- Constraints
- Context
- Resources, including the Functional Specification, if any
- Team and Stakeholders
- Infrastructure
- Practices

} **More on these here** (Charter, Constraints, Context)

} **More on these later** (Team and Stakeholders, Infrastructure, Practices)

aka
Blast Off
Inception
Iteration Zero

Charter

The charter tells everyone why the project is being undertaken.

It is usually short:

- A paragraph or two
- plus some bullets

aka
Goals
Business Case

It may also contains numbers about:

- Revenues
- Units to be sold
- ‘C’-suite information needed to decide whether to proceed.

Constraints

Constraints *and their rationales*

- time
- money
- intermediate targets

Costs

- budgets for development
- unit cost
- memory cost
- etc etc etc

Context

The context document (usually 1~10 pages) captures what the product is supposed to do.

It may also have a *context diagram* that has:

- One big bubble for the system
- External Entities
- Incoming and Outgoing …
- … Data and Control flows

aka
Background
Product Vision
Context
(Diagram)

Resources: Requirements

There’s often a statement of the functions required of the system.

You need to ensure they are all:

- Identified
- Unique
- Coherent
- Unambiguous
- Testable

aka
Requirements
Functional Specification
Functional Requirements
Specification

You also need to learn and regularize the vocabulary.

Resources: Terms

To learn the vocabulary:

- Identify existing
  - documents
  - files
  - listings
- Identify experts

bk.1 ISO Standard xxx
bk.2 User Manual for xxx
bk.3 Code Listing for xxx
bk.4 Configuration file

Sven@myCo.co.se Elevator Expert
Mark@myVendor.com Knows I/O

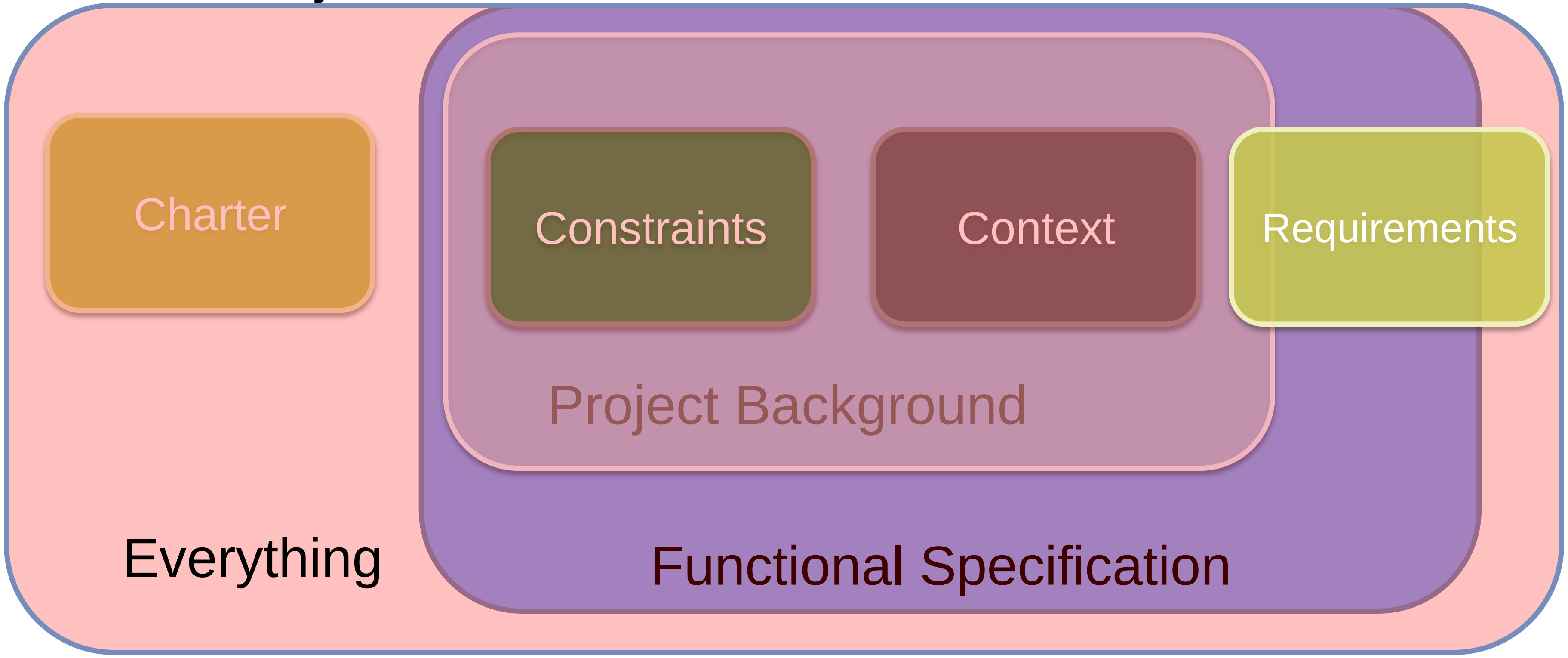
Make an inventory.

It may be helpful to visit the plant or see prior/similar systems

Naming and Packaging

Different organizations have different names for the same thing.

And different organizations package up the elements differently.



Steps

- List what you have and where it is
- Work out how complete it is
- Work out what you need to play back
  what the system does
  to the people who know
  what they want

Workshop

What do you call?

Charter	
Constraints	
Context	
Requirements	

2. Team

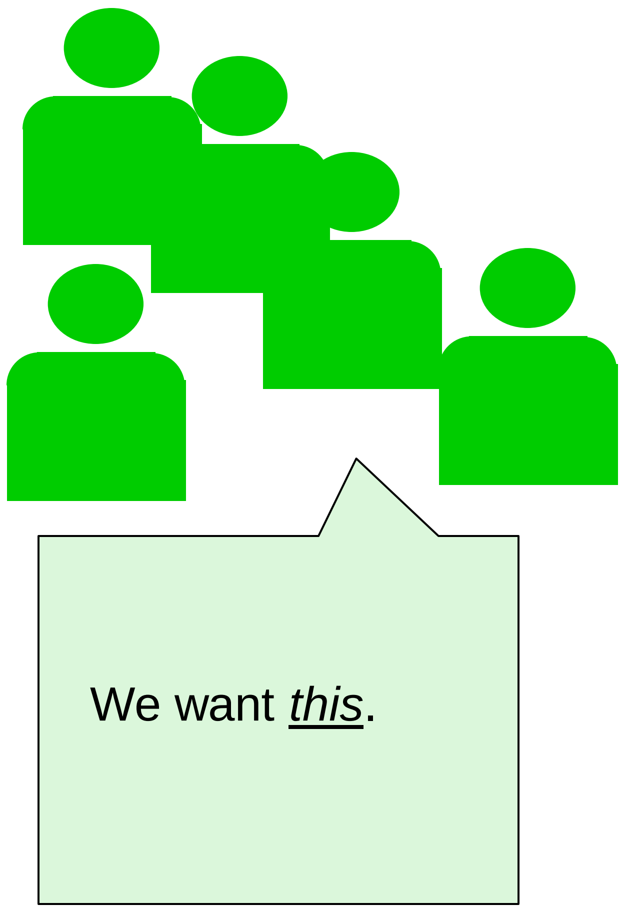
2

Customer Team

The customer team comprises:

- Product management
- Systems engineering
- Acceptance testing
- Business/Product analysts
- Marketing
- Customer service specialist
- etc etc etc

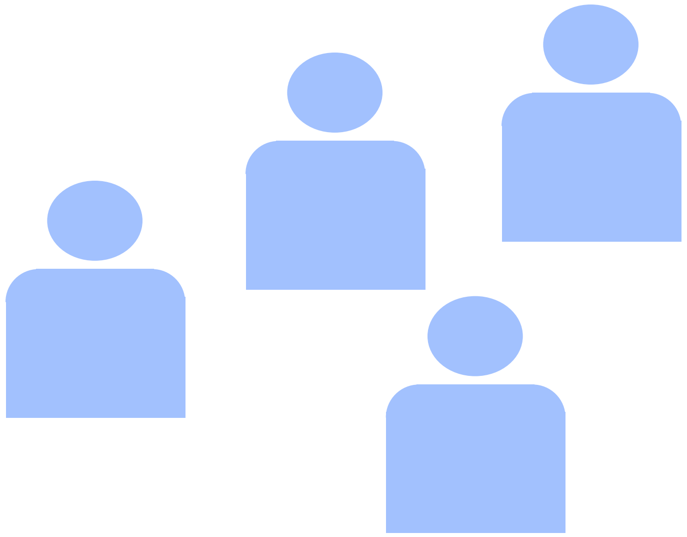
The duty of the customer team is to speak with one voice.



Development Team

The development team comprises:

- software engineers
- hardware engineers
- mechanical engineers
- system engineers



The duty of the development team is to:

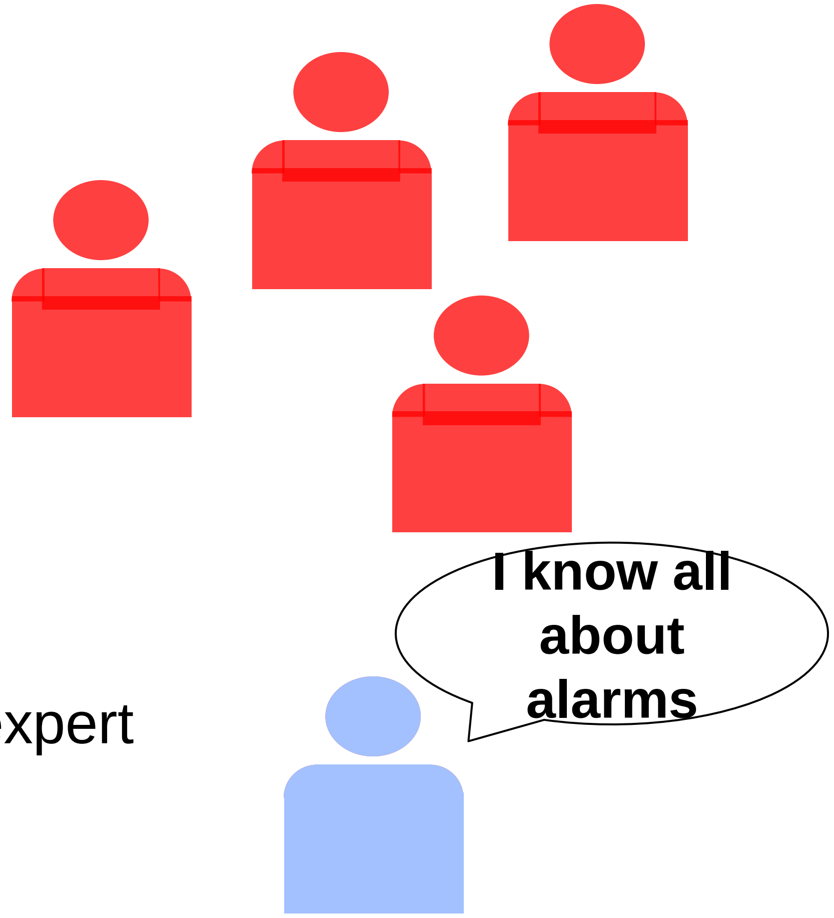
- implement the features demanded by the customer
- advise the customer team on feasibility
  - especially on dependencies

Experts

An expert is someone who knows the technical details of how something works.

They are often:

- Hard to reach
- Assume too much or too little
- A little impatient



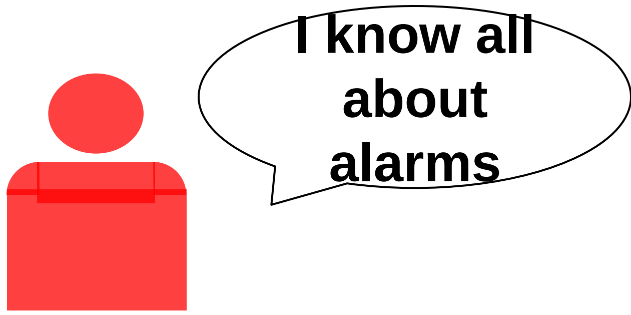
Sometimes *you* are the technical expert

Experts

When you are the expert, you must still write things down, because other people do not have your understanding.

You may, or may not, be:

- Hard to reach
- Assume too much or too little
- A little impatient



It does not change the process.

It just makes it a little easier.

Stakeholders

Stakeholders include anyone who has an interest in the project:

- Regulators
- Competitors
- Other divisions
- Other managers
- People who want you to fail
- People on the team

You need to know who they are and why they care….

… because they are the difference between success and failure

Working Together

The most efficient and effective method of conveying information with and within a development team is face-to-face conversation.

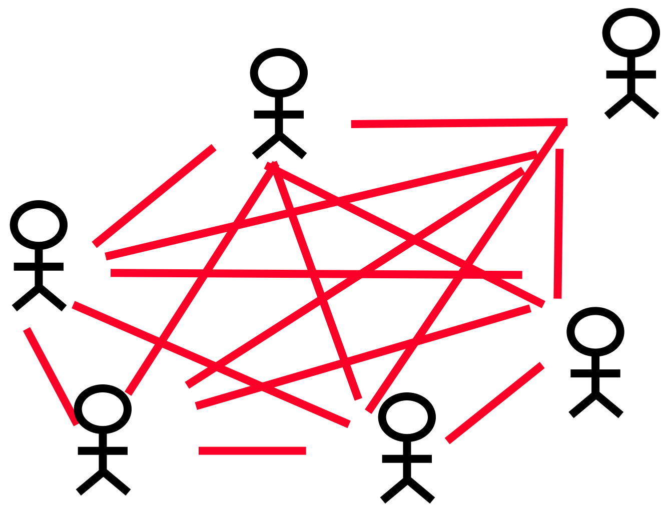
But:

- large development efforts
- geographical distribution
- long-lifecycle products

Large Development Efforts

As communication paths increase difficulties can arise

- pairs
- small teams
- larger teams
- Dunbar's number
- bureaucratic nightmare

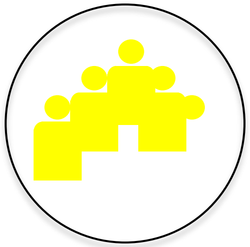
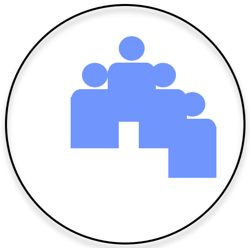


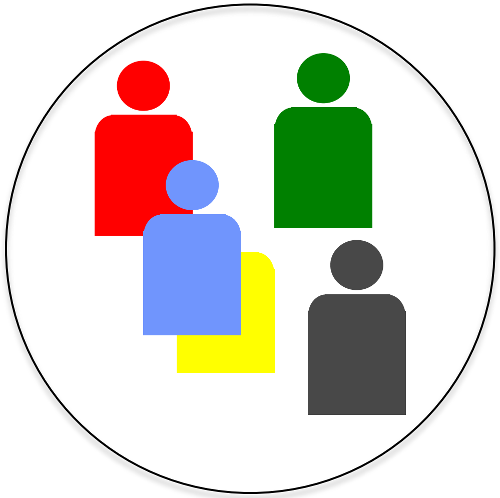
n(n-1) / 2 = 6 * 5 /2 = 15

Successful *large* development efforts are a set of *small* efforts.

Geographical Distribution

- Meet other teams in person
  to get to know them.
  (Best done early.)
- Establish *regular* communication
  for full duplex (a weekly telecon, WebEx etc)
- Establish a whiteboard for
  broad, asynchronous communication
- Schedule meetings (WebEx, F2F, phone)
  to address specific issues



Long-lifecycle Products

A long-lifecycle product will outlive the team.

So think about what they might need:

- Framework
- Ability to learn quickly
- Data files
- System construction
- Build scripts

Build executables (and executable models) where possible!

Modes of Communication

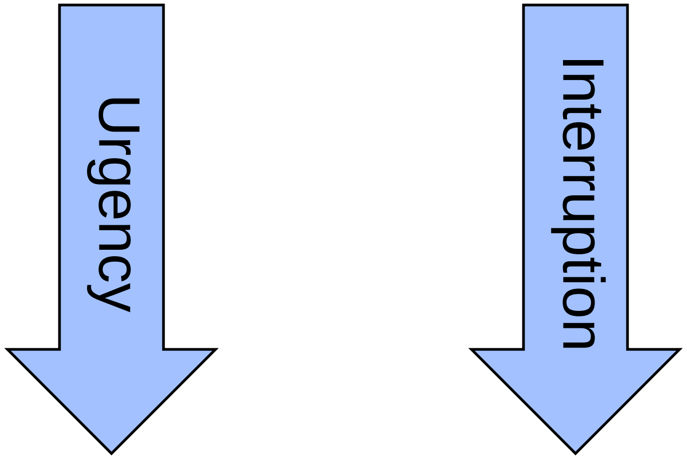
Instant messaging	• Two people • Nearly full duplex • Asynchronous
Group chat	• Multiple people • At will • Asynchronous
Web meetings	• Multiple people • Full multiplex • Scheduled
Shared screens	• Few people • Full duplex • Synchronous
Virtual whiteboard	• Multiple people • At will • Asynchronous

Priority

You don’t want to interrupt other peoples’ thoughts.

But you do need answers.

- Email
- IM / SMS
- Ring desk phone
- Ring cell phone



Workshop

Which services do you use to:

Meet together in the different locations?	
Ensure *regular* communication?	
Let people know what you’re doing?	
Ask questions to the team?	

3. Communication

3

Face-to-Face Communication

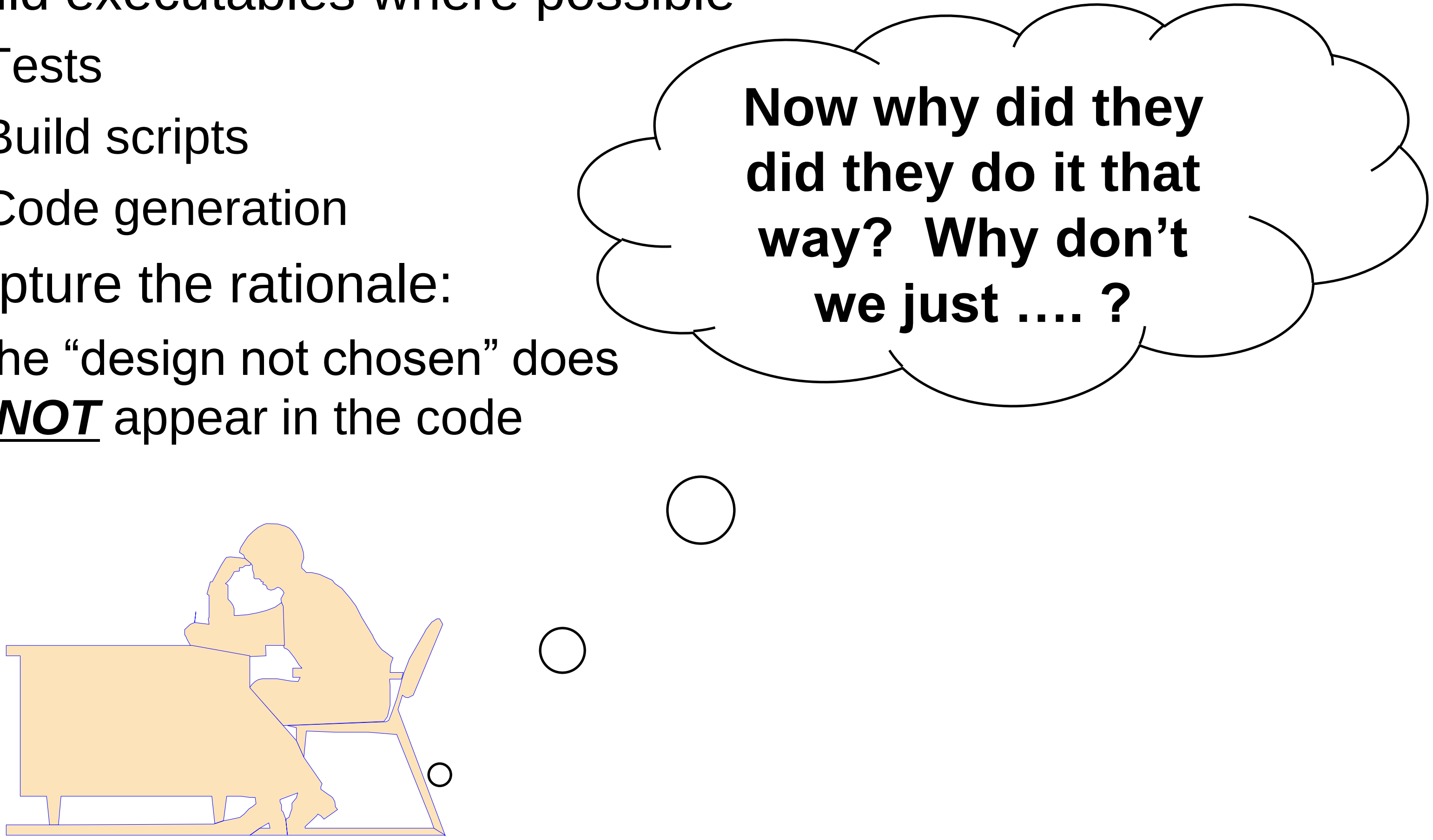
The most efficient and effective method of conveying information with and within a development team is face-to-face conversation.

But:

- long-lifecycle products
- finding people at the right time
- too many things to remember

Long-lifecycle Products

- Define Theory of Operation
- Build executables where possible
  - Tests
  - Build scripts
  - Code generation
- Capture the rationale:
  - the "design not chosen" does ***NOT*** appear in the code


Now why did they did they do it that way? Why don't we just …. ?

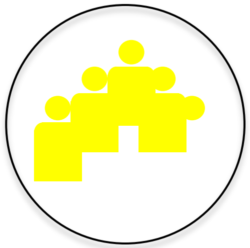
Finding People

Office	Cube & Conf	War Room	Team Room
Private	Not private	Choose to talk	Communal
Quiet	Noisy	Separated	Protocols
Open door vs Closed door	‘Door’ always open	Dedicated to common use	Where do you think?
Scheduled	Serendipitous	Both	Constant

Multiple Locations

Geographical distribution requires other techniques:

- Regular teleconferences (across time zones)
- Reduce divergence
- Maintain relationships
- E-mail is a "pull" medium
  - Use e-mail for technical stuff only.
  - *Never* use e-mail for emotional subjects.
  - Sleep on "difficult" e-mails.
- A single task list to maintain momentum.



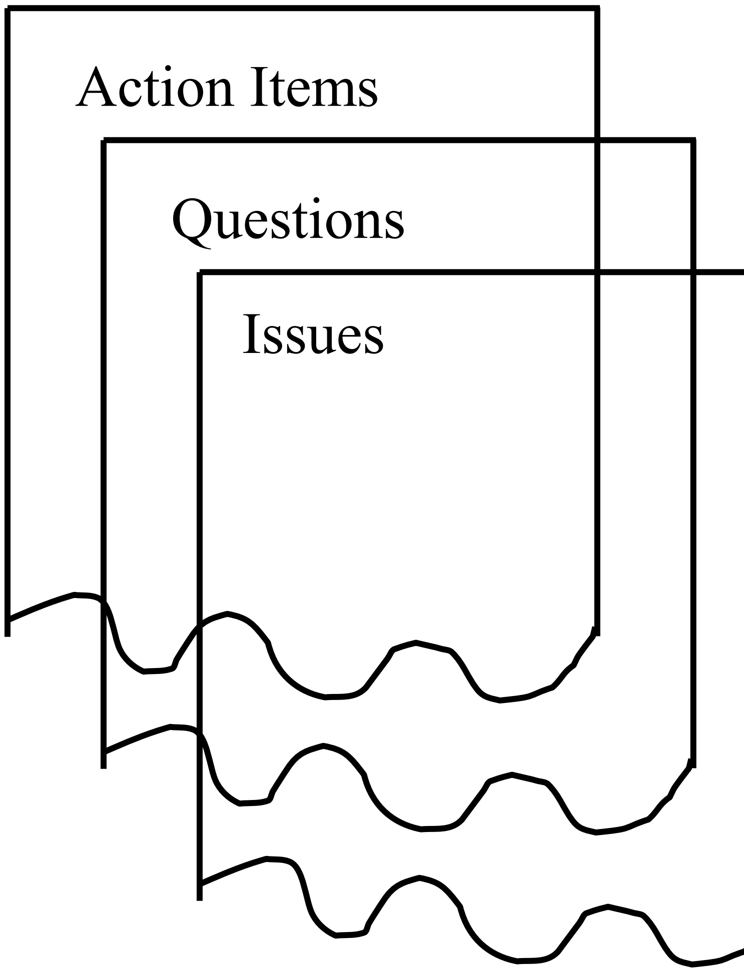
Too Many Things to Remember

You will need to write down at least:

- Action Items
- Questions for experts
- Issues

- Background documents
- Technical notes

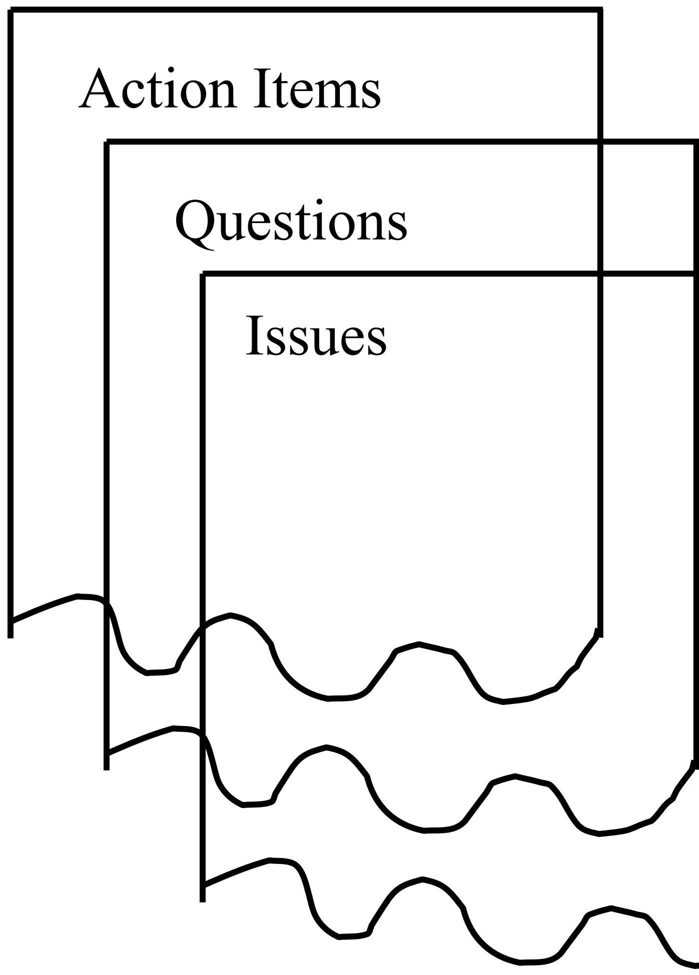
among others.



Tracking

You need to track ongoing items:

- Whose responsibility is it? ("the owner")
- What state is it in?
  - open
  - pending reply from expert
  - in progress (current sprint)
  - closed



Requirements

As you read background documents, you will encounter:

- Well-defined requirements (even if people have slightly different interpretations of the exact meaning)
- Multiple requirements in one
- Mysteries

For "mystery requirements",

- Go back to the authors and have them rewrite it

Requirement 23 has been factored:
23.A
23.B
and 23.C

For "multiples" ,

- Go back to the authors and have them rewrite it
- If that fails, factor it into "subrequirements"

Terms

As you read background documents, you will encounter:

- Well-defined terms, often defined in the relevant standards or texts
- Well-understood terms (even if people have slightly different interpretations of the exact meaning)
- Mysteries

For “mystery terms”,

- List them
- Identify (near) synonyms
- List questions

Some terms are “abused.” Everyone has their own (inconsistent) definitions.

Use “Fred,” instead.

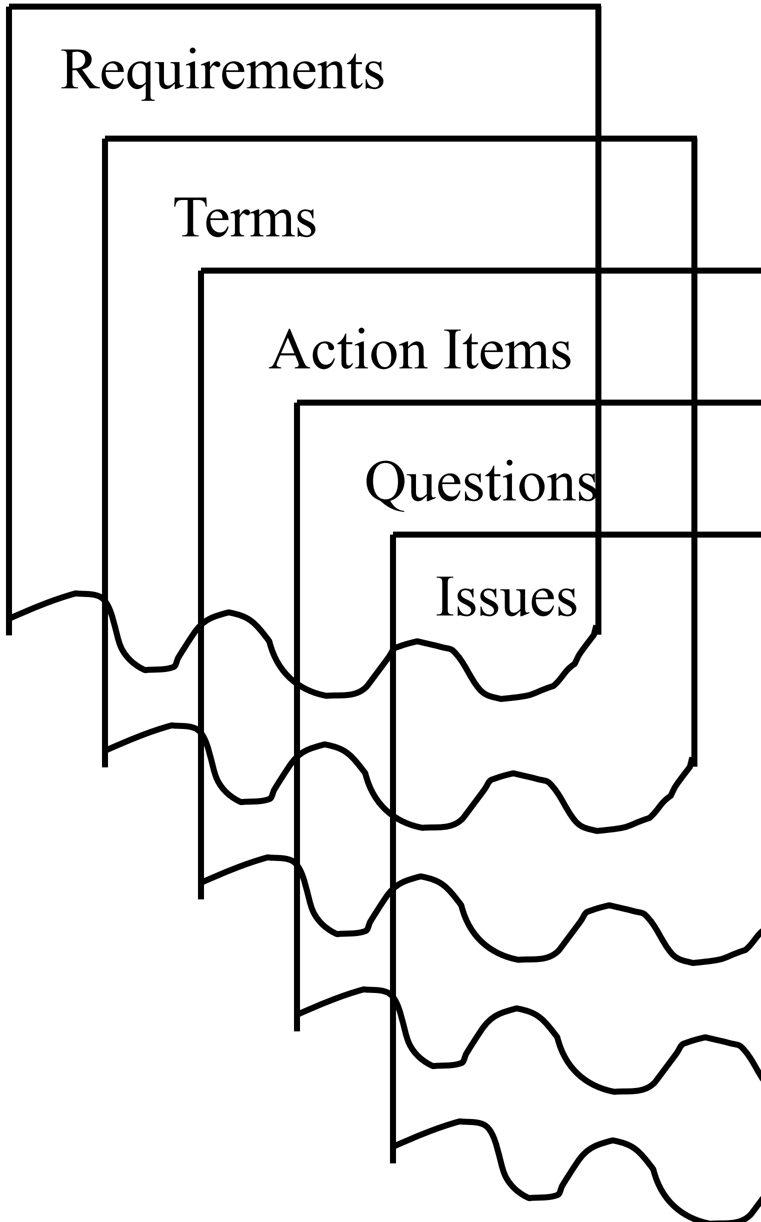
Writing It Down

Add any

- Requirements
- Terms

to your other documents.

Do not attempt the rewrite the background documents.



Infrastructure

You will need document infrastructure, but it must be:

- cheap,
- incremental,
- only “as-needed”
- low learning curve
- easy to use

A wiki enables asynchronous conversations.

Transparency

Information needs to be public, because:

- It belongs to the company, not you
- It is expensive to find information in locked drawers
- Duplication is less likely
- Others can usually improve on what you have
- It’s easier to start with something that exists already

Workshop

What do you call …?

Requirements	
Terms	
Action Items	
Questions	
Issues	

Workshop

And where do you keep them?

Requirements	
Terms	
Action Items	
Questions	
Issues	

4. Practices

4

Recording

You need to come to agreement on how you write down:

- Requirements
- Terms

- Action Items
- Questions for experts
- Issues

- Technical notes

- How detailed?
- How will they managed?
- Who “owns” the collection?

Daily Stand Up

A *short* daily coordination meeting for developers(*) covering:

- What did I do yesterday?
- What do I plan to do today?
- What is in my way?

Take problem solving and tangents offline.

Focus on meeting team goals.

(*) You'll also need to decide how to meet with customers etc

Weekly Sit Down

Or perhaps a weekly group meeting is preferable.

- Start on time—no matter what
- End on time (or before)—no matter what
- Don’t repeat what someone missed—no matter what

Take problem solving and tangents offline.

Focus on meeting team goals.

Other Team Practices

The team should agree on their practices (eg):

- Test-driven development (i.e., when to run tests)
- Continuous review
- Continuous integration
- Build procedures etc
- How long will an iteration be?
- When will you meet?
- Learning goals:
    - What additional skills will you have at the end?

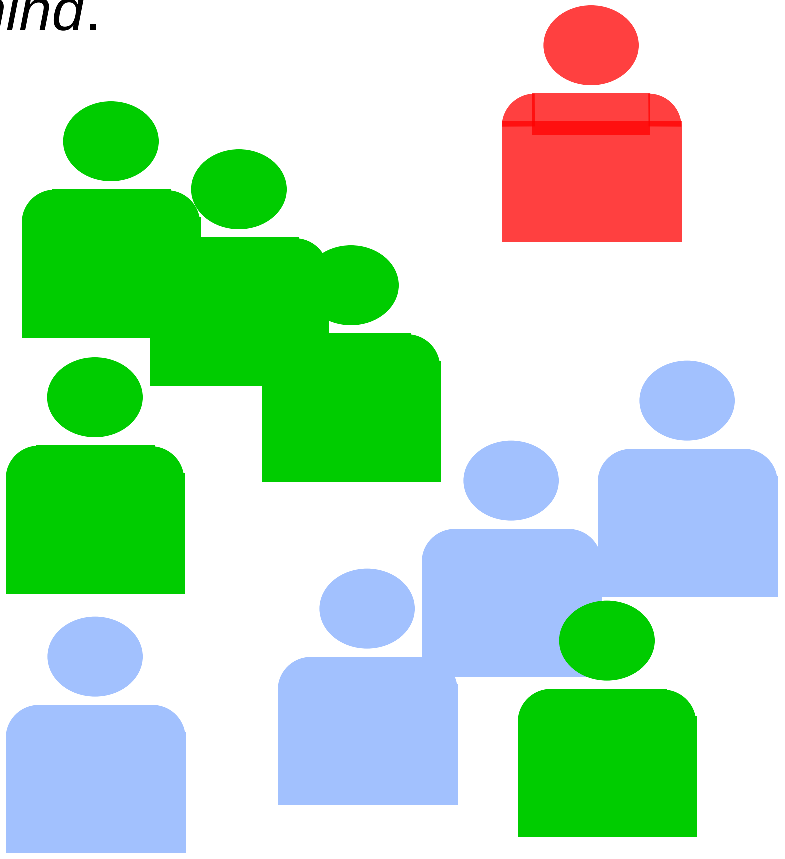
Collective Ownership

Collective ownership means the artifacts look as if they were *produced by a single mind*.

This means:

- standards
- anyone can change anything, anytime
- everyone is responsible for the final product

You'll need to learn about other disciplines.



Revisit the Team

Do we have the:

- right stakeholders?
  - regulatory?
  - other business units?
- right technical skills?
  - OS?
  - domain knowledge
- right customers and experts?
  - are the customers empowered to make decisions?
  - do you need more expert knowledge?

Workshop

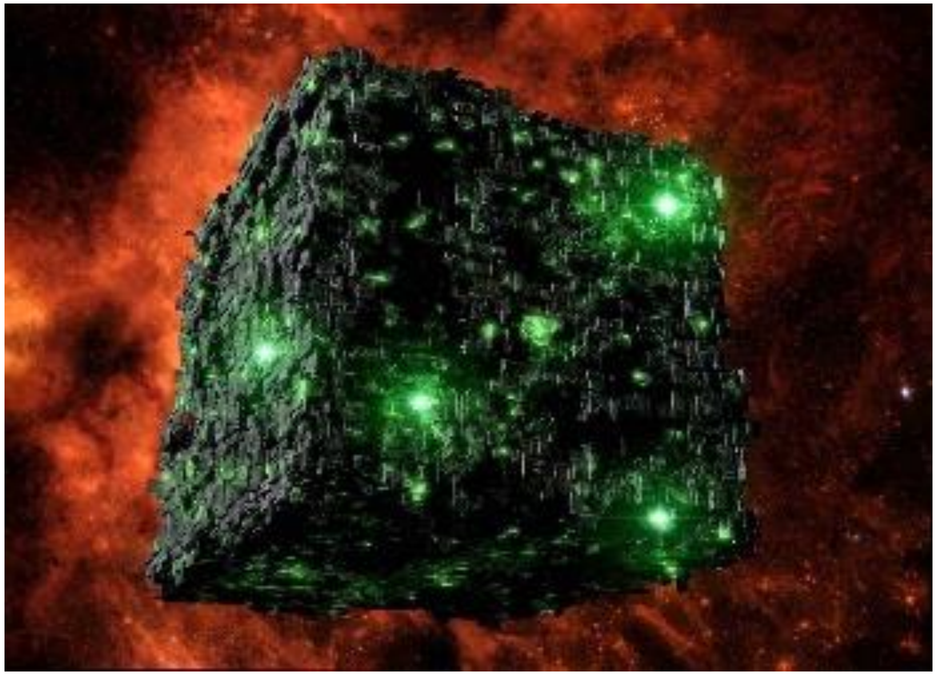
List your team practices.

5. Assimilation

Resistance is Futile

Now that you know:

- Who is on the team
- What it is you're trying to do
- How you'll work together
- How you'll record information, and
- All the resources….



it's time to go to work!

How will you assimilate the materials?

Find Requirements

Read the specification to find requirements.

They must be:

- Identified
- Unique
- Coherent
- Unambiguous
- Testable

Find & Identify Your Requirements

Find each requirement and ensure it’s identified.

Req’t 1: The door must open if it is obstructed.

Req’t 2: A passenger must be able to get an elevator to a floor. ✗

Req’t : A passenger must be able to open the door. ✗

Coherent and Unambiguous

Each requirement must be coherent and unambiguous.

Req’t 1: The door must open if it is obstructed.

Req’t 2.A: A passenger must be able to order the elevator he occupies to a floor.

Req’t 2.B: A passenger must be able to request an elevator moving in a specified direction to a floor.

Req’t 3: A passenger must be able to open the door.

Testable

Each requirement should be testable.

You must be able to know if the requirement has been met.

Req’t 10: The administrator must be able to define a mode. ✗

Req’t 10: The administrator must be able to switch between two modes. Morning mode brings all unused elevators to the ground floor. Normal mode leaves them where they are.

Req’t 11: The elevator must be able to service many floors ✗

Req’t 11: The elevator must be able to service a maximum of 100 floors.

Testable

Each requirement must be testable. You must be able to say:

- what is true before the requirement executes
- what is true after the requirement has executed

Req’t 1: The door must open if it is obstructed.
Pre: Obstruction in door
Post: Door open

Req’t 2: A passenger must be able to order the elevator he occupies to a floor.
Pre: None
Post: Order to move elevator to floor queued

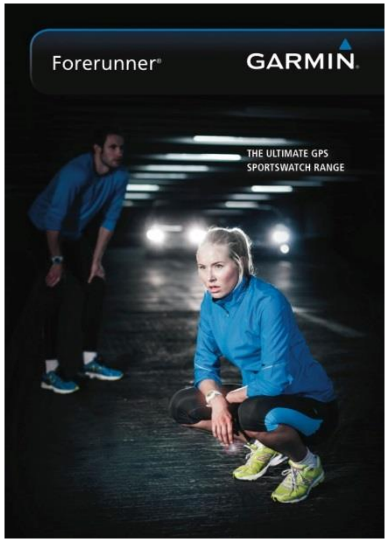
Workshop

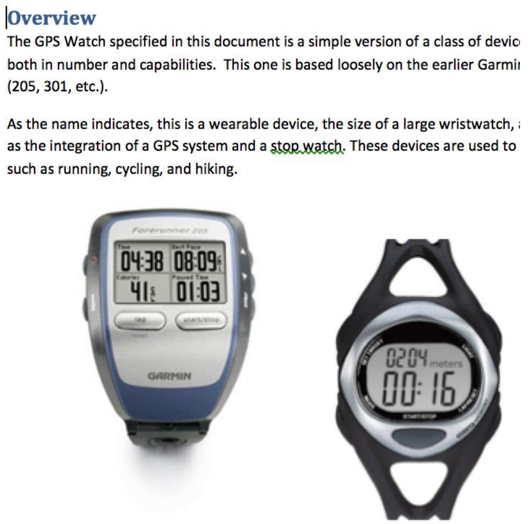
Read the description for the case study.

Handouts

Identify the requirements, and decide if they are:

- Identified
- Unique
- Coherent
- Unambiguous
- Testable





Overview

The GPS Watch specified in this document is a simple version of a class of devic
both in number and capabilities. This one is based loosely on the earlier Garmi
(205, 301, etc.).

As the name indicates, this is a wearable device, the size of a large wristwatch,
as the integration of a GPS system and a stop watch. These devices are used to
such as running, cycling, and hiking.

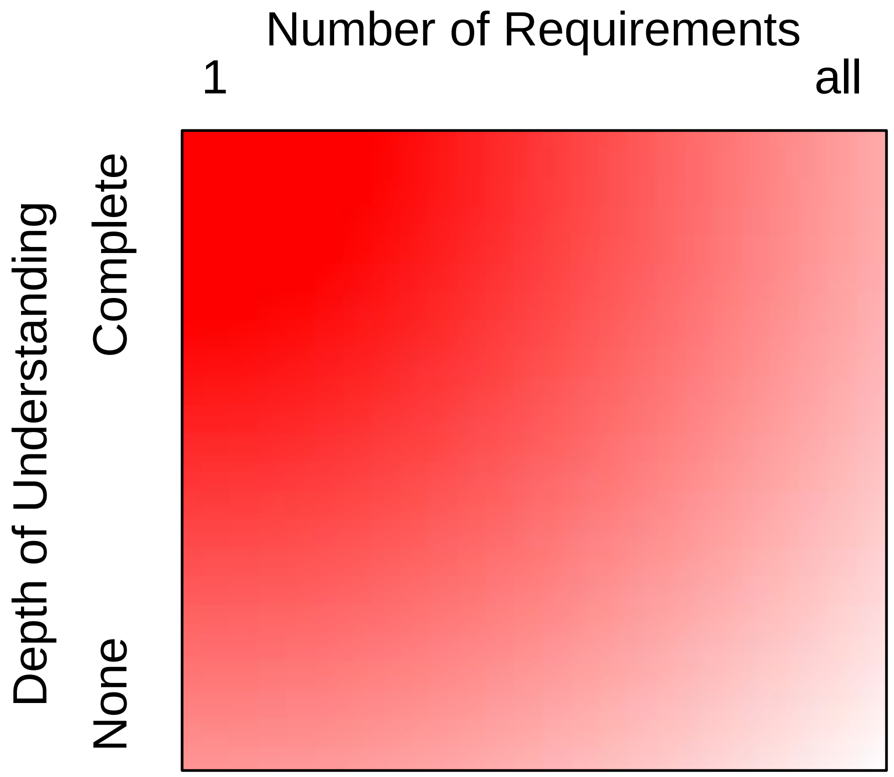
If not, ask your customer.

6. Process

6

How Wide?

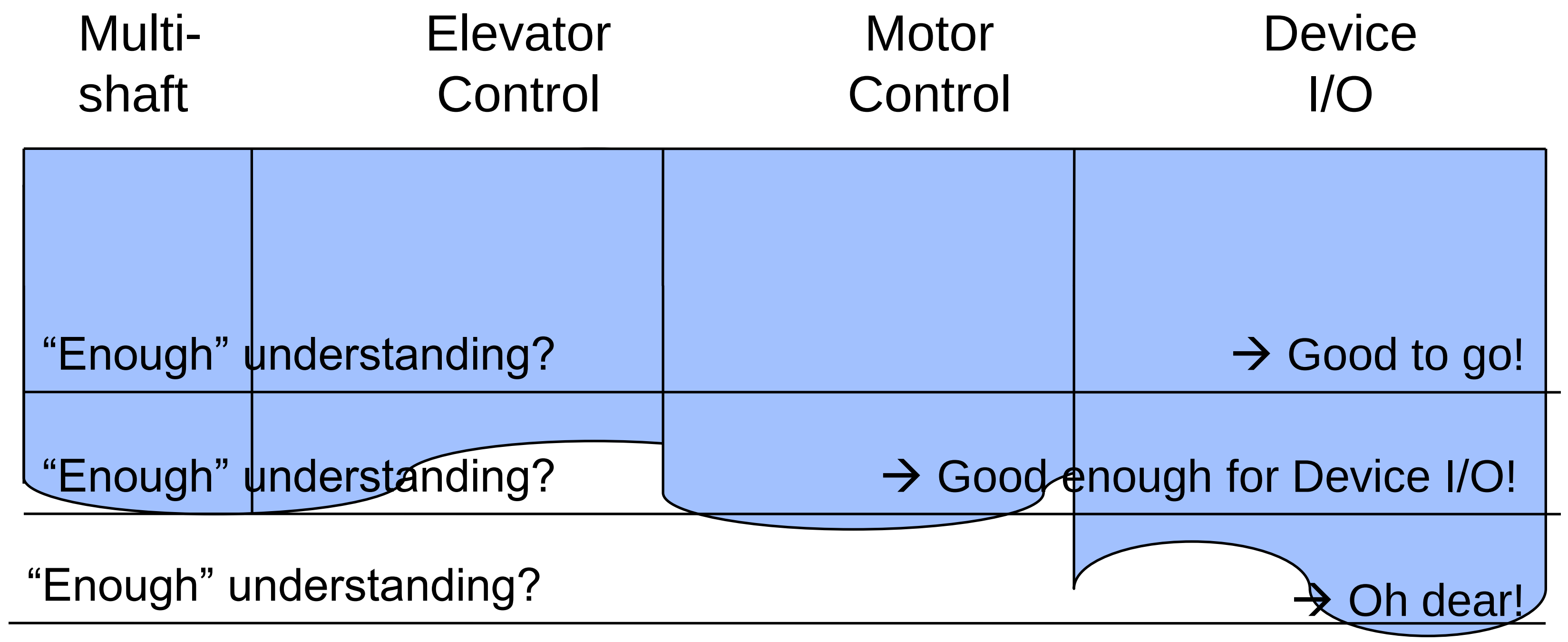
Do you need all the requirements, before you go into detail?



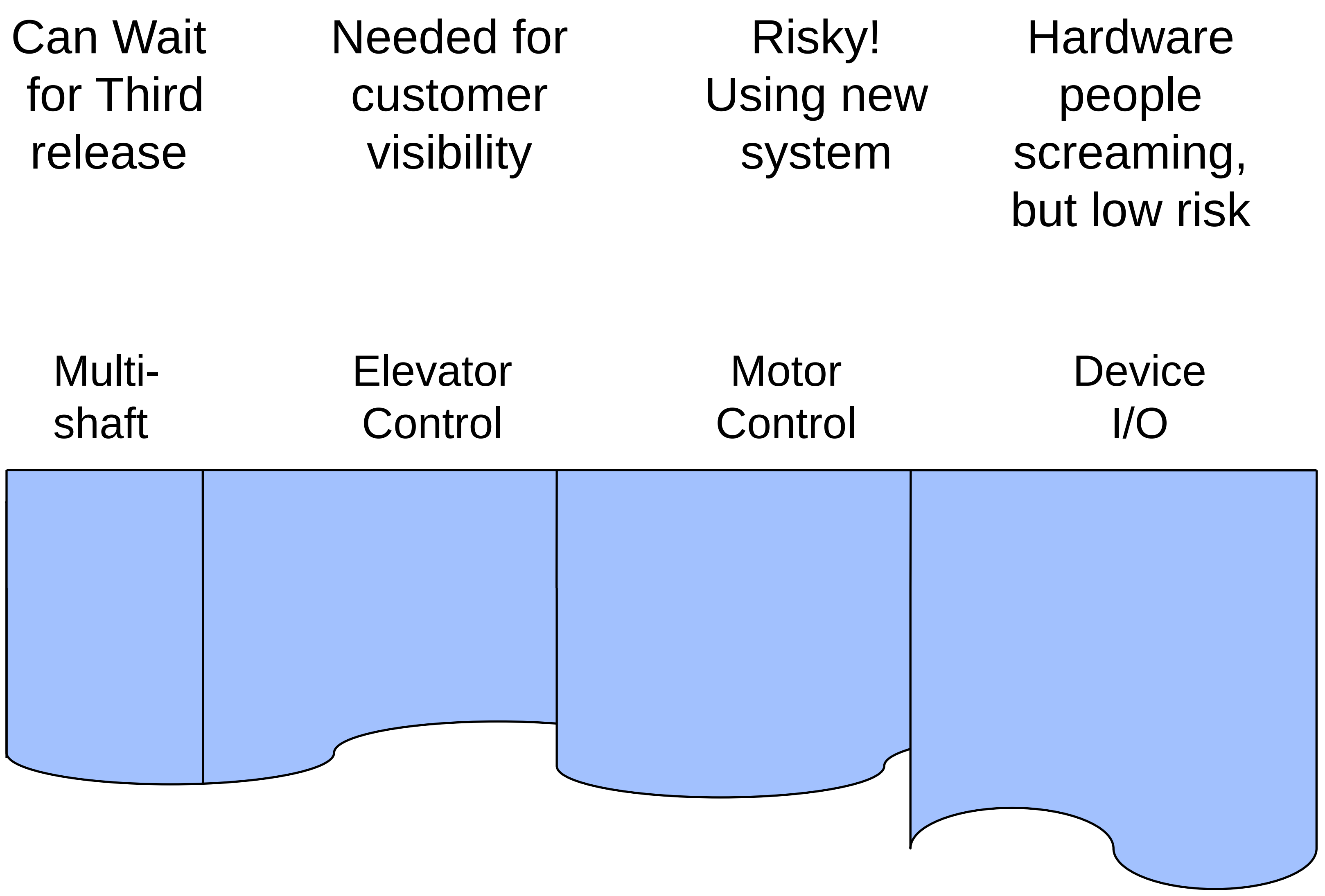
Depth of Understanding

Determine your depth of understanding.

Your mileage may vary!

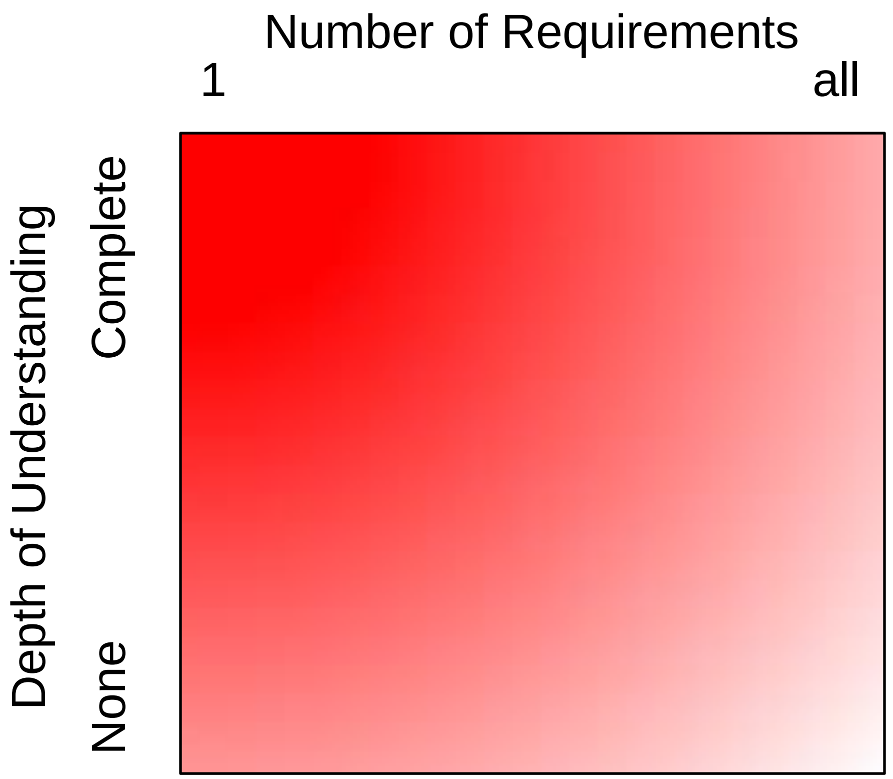


Risk et al.

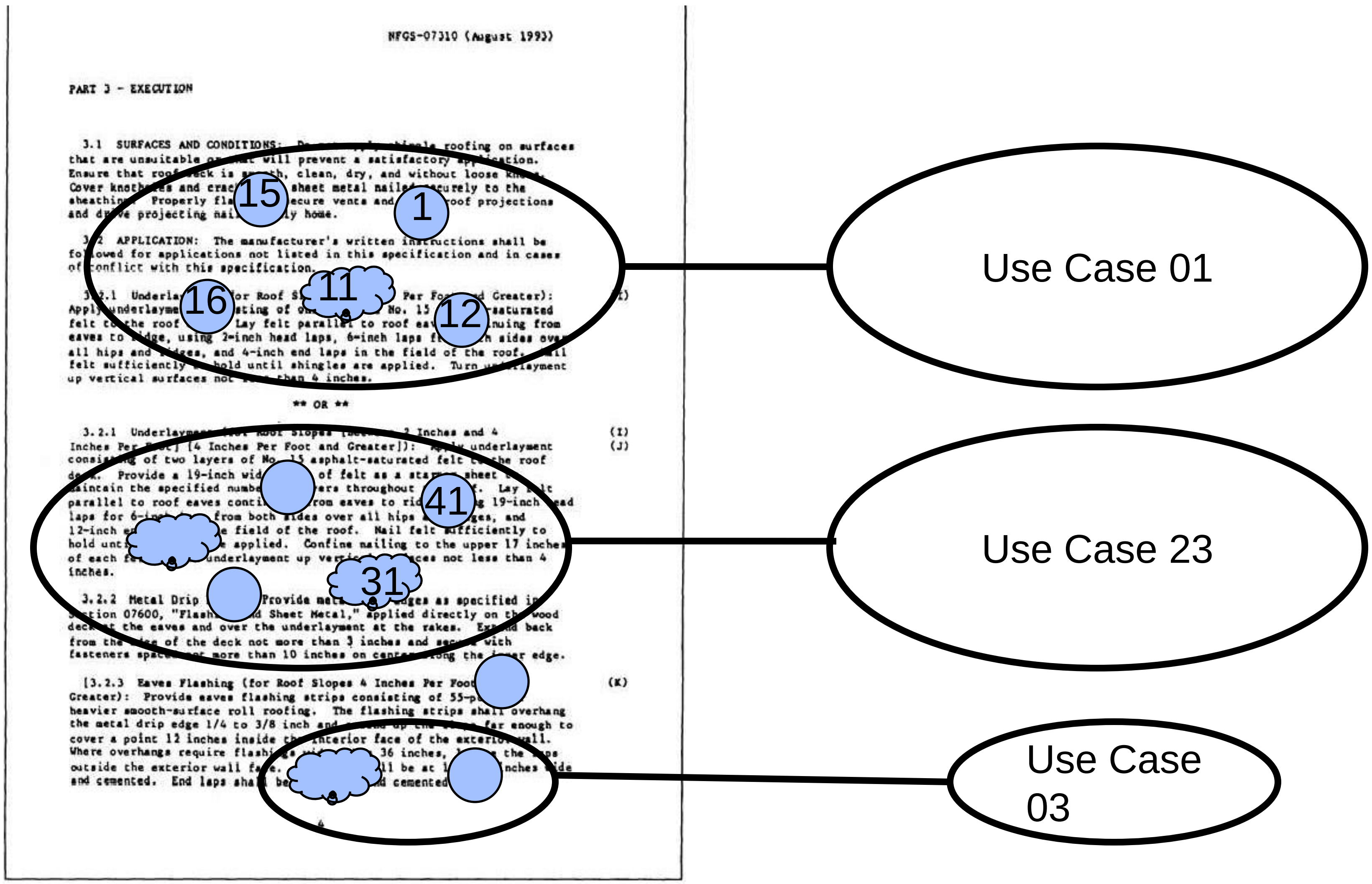

Can Wait for Third release
Needed for customer visibility
Risky! Using new system
Hardware people screaming, but low risk
Multi-shaft
Elevator Control
Motor Control
Device I/O

How Wide?

- Go wide initially
- Focus on specific areas, according to:
  - Risk
  - Staffing
  - Etc
- Learn from going deep
- Go wide again



Details



NFGS-07310 (August 1993)

PART 3 - EXECUTION

3.1 SURFACES AND CONDITIONS: [illegible] shingle roofing on surfaces that are unsuitable or [illegible] will prevent a satisfactory [illegible]cation. Ensure that roof [illegible]ck is [illegible]th, clean, dry, and without loose kn[illegible]. Cover knoth[illegible]es and crac[illegible] sheet metal nailed [illegible]curely to the sheathing. Properly fla[illegible]ecure vents and [illegible] roof projections and d[illegible]ve projecting nail[illegible]ly home.

3.2 APPLICATION: The manufacturer's written instructions shall be followed for applications not listed in this specification and in cases of conflict with this specification.

3.2.1 Underlay[illegible]for Roof S[illegible] Per Fo[illegible]d Greater): (I) Apply underlayme[illegible]sting of o[illegible] No. 15 [illegible]-saturated felt to the roof [illegible] Lay felt parallel to roof eav[illegible]nuing from eaves to ridge, using 2-inch head laps, 6-inch laps f[illegible]h sides over all hips and [illegible]dges, and 4-inch end laps in the field of the roof. [illegible]il felt sufficiently [illegible] hold until shingles are applied. Turn u[illegible]layment up vertical surfaces not [illegible] than 4 inches.

** OR **

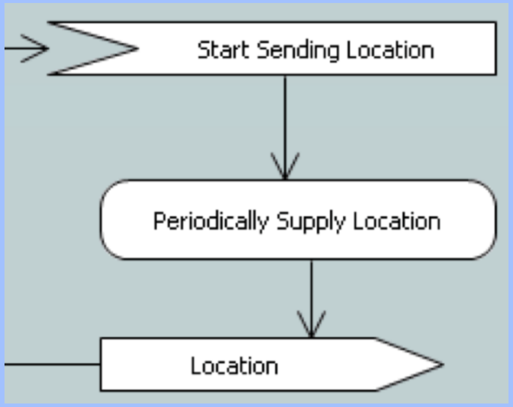
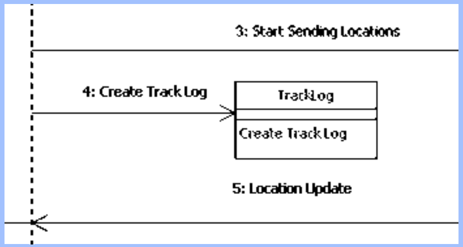
3.2.1 Underlaym[illegible] Roof Slopes [[illegible] 2 Inches and 4 Inches Per [illegible]] [4 Inches Per Foot and Greater]): [illegible] underlayment consis[illegible] of two layers of No. 15 asphalt-saturated felt [illegible] the roof de[illegible]. Provide a 19-inch wid[illegible] of felt as a star[illegible] sheet [illegible]aintain the specified numbe[illegible]ers throughout [illegible]f. Lay [illegible]lt parallel to roof eaves conti[illegible]rom eaves to rid[illegible]g 19-inch [illegible]ead laps for 6-in[illegible] from both sides over all hips [illegible]ges, and 12-inch e[illegible]e field of the roof. Nail felt sufficiently to hold unti[illegible] applied. Confine nailing to the upper 17 inches of each fe[illegible] underlayment up verti[illegible]ces not less than 4 inches. (I) (J)

3.2.2 Metal Drip [illegible] Provide met[illegible]dges as specified in [illegible]ction 07600, "Flash[illegible]d Sheet Metal," applied directly on the wood deck [illegible] the eaves and over the underlayment at the rakes. Ex[illegible]nd back from the [illegible] of the deck not more than 3 inches and sec[illegible] with fasteners spac[illegible] more than 10 inches on cent[illegible]ong the [illegible]r edge.

[3.2.3 Eaves Flashing (for Roof Slopes 4 Inches Per Foot [illegible] Greater): Provide eaves flashing strips consisting of 55-p[illegible] heavier smooth-surface roll roofing. The flashing strips shall overhang the metal drip edge 1/4 to 3/8 inch and [illegible] far enough to cover a point 12 inches inside the interior face of the exterior wall. Where overhangs require flashi[illegible] 36 inches, [illegible] the [illegible]ps outside the exterior wall fa[illegible]e. [illegible]ll be at l[illegible]nches [illegible]ide and cemented. End laps sha[illegible] be[illegible]d cemented. (K)

4

Specification

Actor: Trigger: Pre-conditions: Post-conditions: Scenario: Requirements:	 Start Sending Location Periodically Supply Location Location	 3: Start Sending Locations 4: Create Track Log TrackLog Create Track Log 5: Location Update

B: Functional Behavior

7. Use Cases

7

Use Cases

A *use case* says how a role uses a system to meet some goal.

aka “usage case”

Examples:

- A passenger requests elevator
- A system administrator sets elevator mode
- At 08:00, free elevators return to the ground floor.
- When the door reaches the floor, the door opens

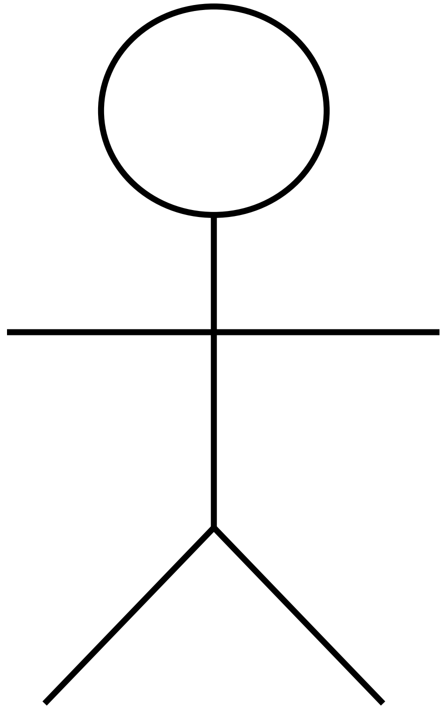
Actors

An actor is anything that interacts with the system:

- people
- machines and sensors
- other systems
- timers

We name the role, not the person or thing.

- Fred → Administrator
- Fred → Passenger
- Mary → Passenger



Interaction

An interaction between an actor and system is shown as an association, thus:

It crosses the system boundary and consists of:

- data flows
- control flows
- or a complex interaction comprising multiple flows

Other relationships exist within the system boundary.

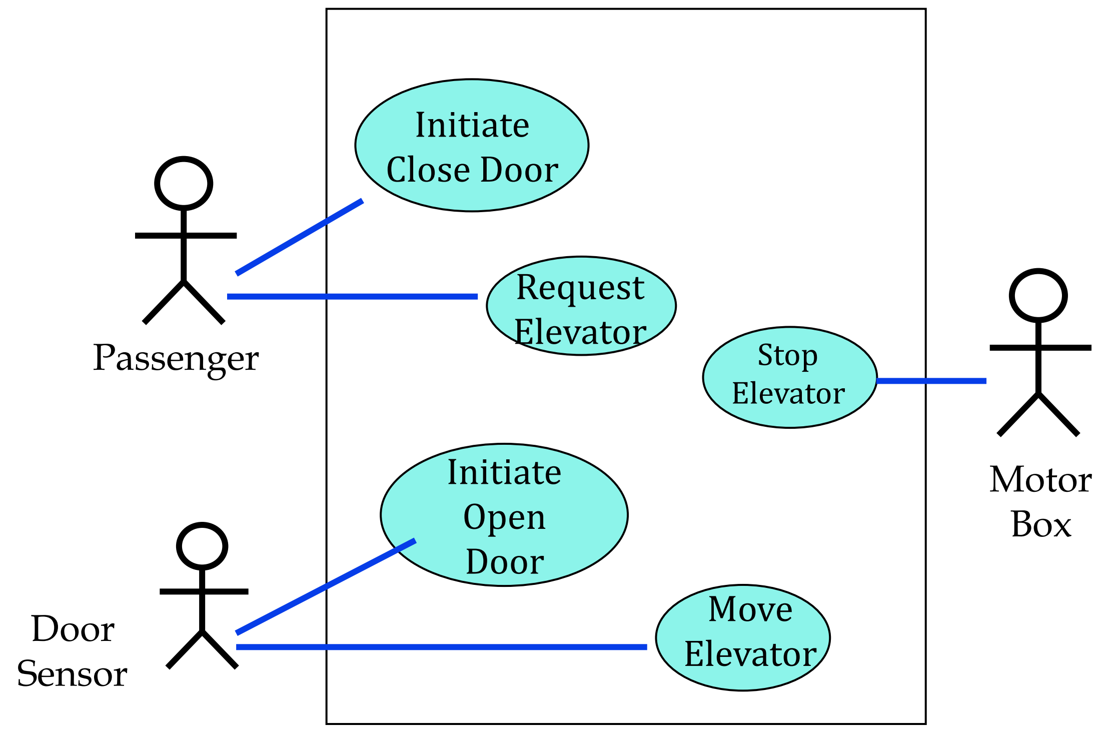
Use Case

The *use case* is everything the system does for the actor.

- A passenger requests elevator →
  Bring an elevator to the requesting floor
- A system administrator sets elevator mode →
  Change mode from ‘Normal’ to ‘Evening’
- At 08:00, free elevators return to the ground floor →
  Change mode from ‘Evening’ to ‘Morning’
- When the door reaches the floor, the door opens →
  Open the door

It can include complex interactions back and forth

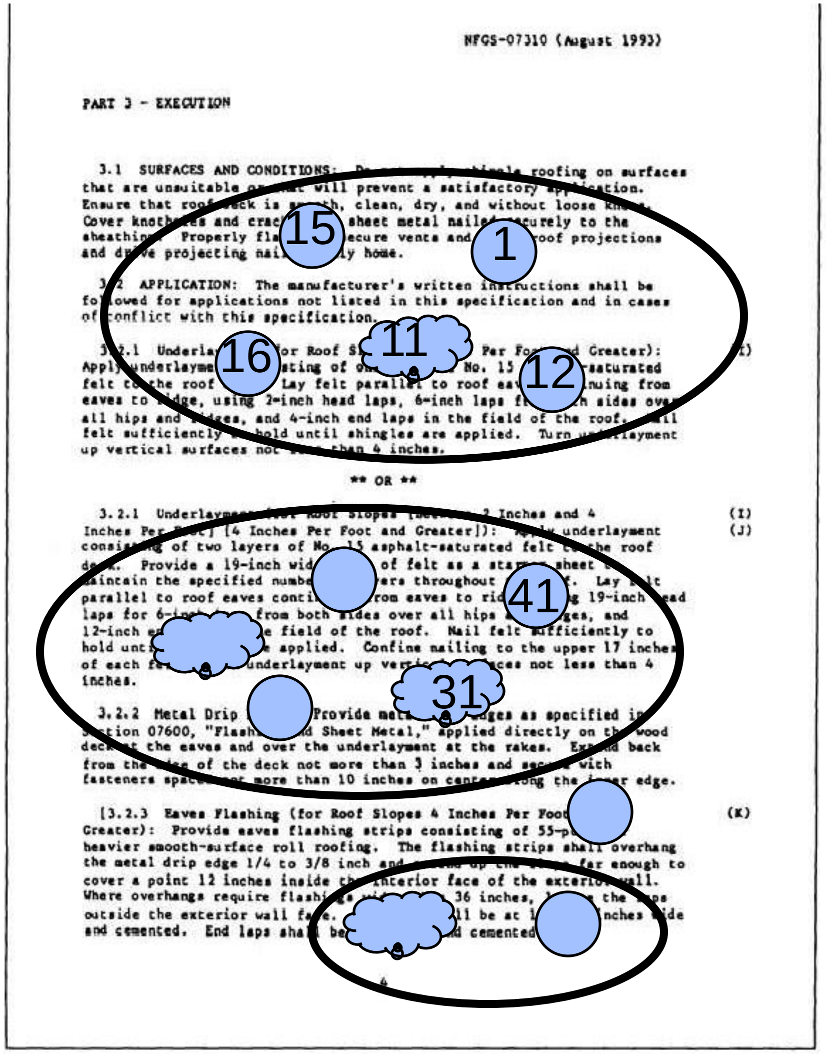
Use Case Diagram



8: Finding Use Cases

8

Read the Spec.!


15
1
16
11
12
41
31

PART 3 - EXECUTION

3.1 SURFACES AND CONDITIONS: [illegible] shingle roofing on surfaces that are unsuitable or [illegible] will prevent a satisfactory application. Ensure that roof [illegible] is [illegible]h, clean, dry, and without loose kn[illegible]. Cover knoth[illegible]es and crac[illegible] sheet metal nailed [illegible]curely to the sheathing. Properly fla[illegible]ecure vents and [illegible] roof projections and d[illegible]ve projecting nai[illegible]ly home.

3.2 APPLICATION: The manufacturer's written instructions shall be followed for applications not listed in this specification and in cases of conflict with this specification.

3.2.1 Underla[illegible] for Roof S[illegible] Per Fo[illegible]d Greater): (I)
Apply underlayme[illegible]sting of o[illegible] No. 15 [illegible]-saturated felt to the roof [illegible] Lay felt parallel to roof eav[illegible]nuing from eaves to ridge, using 2-inch head laps, 6-inch laps f[illegible]h sides over all hips and ridges, and 4-inch end laps in the field of the roof. [illegible]il felt sufficiently [illegible] hold until shingles are applied. Turn underlayment up vertical surfaces not [illegible] than 4 inches.

** OR **

3.2.1 Underlayment [illegible] Roof Slopes [[illegible] 2 Inches and 4 Inches Per [illegible]] [4 Inches Per Foot and Greater]): [illegible] underlayment consisting of two layers of No. 15 asphalt-saturated felt to the roof deck. Provide a 19-inch wid[illegible] of felt as a star[illegible] sheet [illegible] maintain the specified numbe[illegible]ers throughout [illegible]f. Lay felt parallel to roof eaves conti[illegible]rom eaves to rid[illegible]g 19-inch head laps for 6-inc[illegible] from both sides over all hips [illegible]ges, and 12-inch e[illegible]e field of the roof. Nail felt sufficiently to hold unti[illegible]e applied. Confine nailing to the upper 17 inches of each fe[illegible] underlayment up verti[illegible]ces not less than 4 inches. (I) (J)

3.2.2 Metal Drip [illegible] Provide met[illegible]dges as specified in Section 07600, "Flash[illegible]d Sheet Metal," applied directly on the wood deck at the eaves and over the underlayment at the rakes. Extend back from the [illegible] of the deck not more than 3 inches and secure with fasteners spac[illegible] more than 10 inches on center along the inner edge.

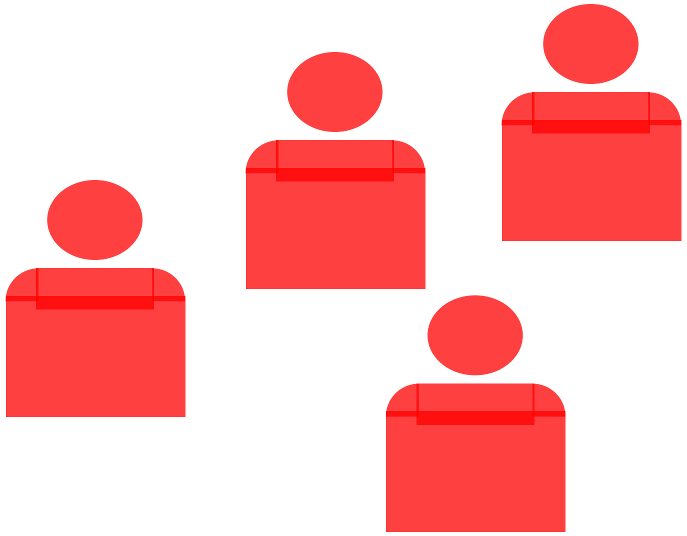
[3.2.3 Eaves Flashing (for Roof Slopes 4 Inches Per Foot [illegible] Greater): Provide eaves flashing strips consisting of 55-p[illegible] heavier smooth-surface roll roofing. The flashing strips shall overhang the metal drip edge 1/4 to 3/8 inch and [illegible] far enough to cover a point 12 inches inside the interior face of the exterior wall. Where overhangs require flashi[illegible] 36 inches, [illegible] the laps outside the exterior wall fa[illegible]e. [illegible]ll be at l[illegible]nches wide and cemented. End laps shal[illegible] be[illegible]d cemented [illegible] (K)

4

Identify Personnel

The people that know the subject matter best are usually the experts and the customers.

Invite them to the initial sessions, and
be prepared to ask them for more detail later.



Blitz

A *blitz* is a technique for getting started.

There are no wrong answers.

- We don’t categorize
- We don’t organize
- We don’t evaluate
- We just enumerate

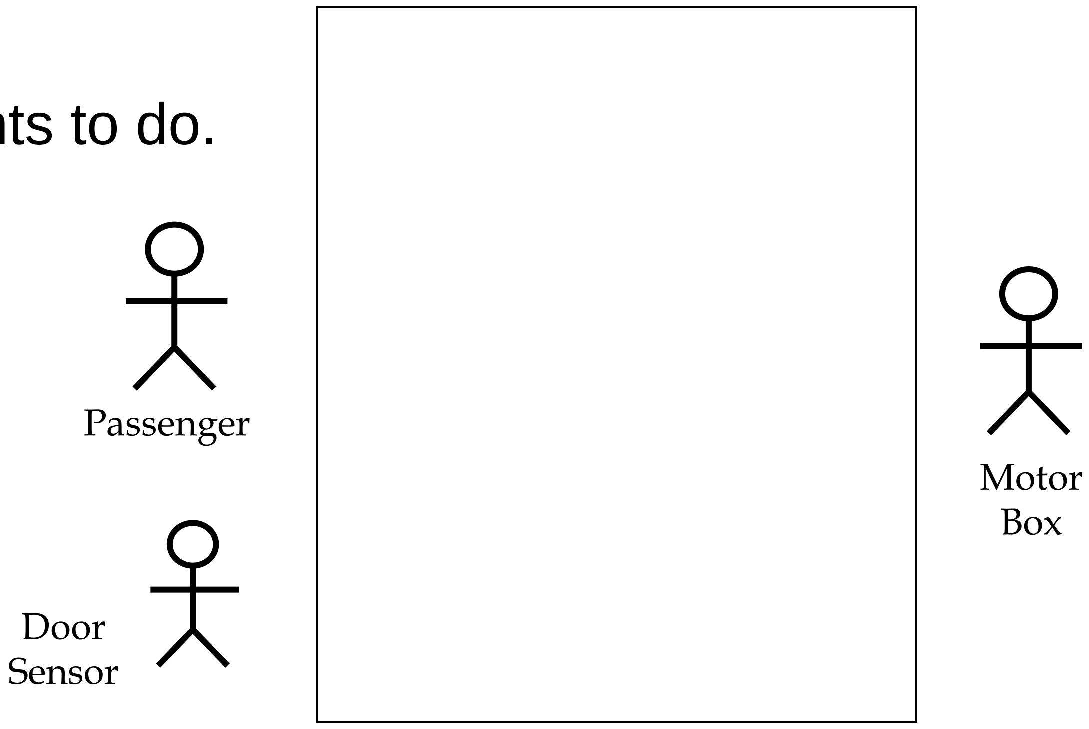
The purpose is to provide a starting point.

Actors

Who or what interacts with the system?

- People (as roles)
- Machines and devices
- Other systems
- Time

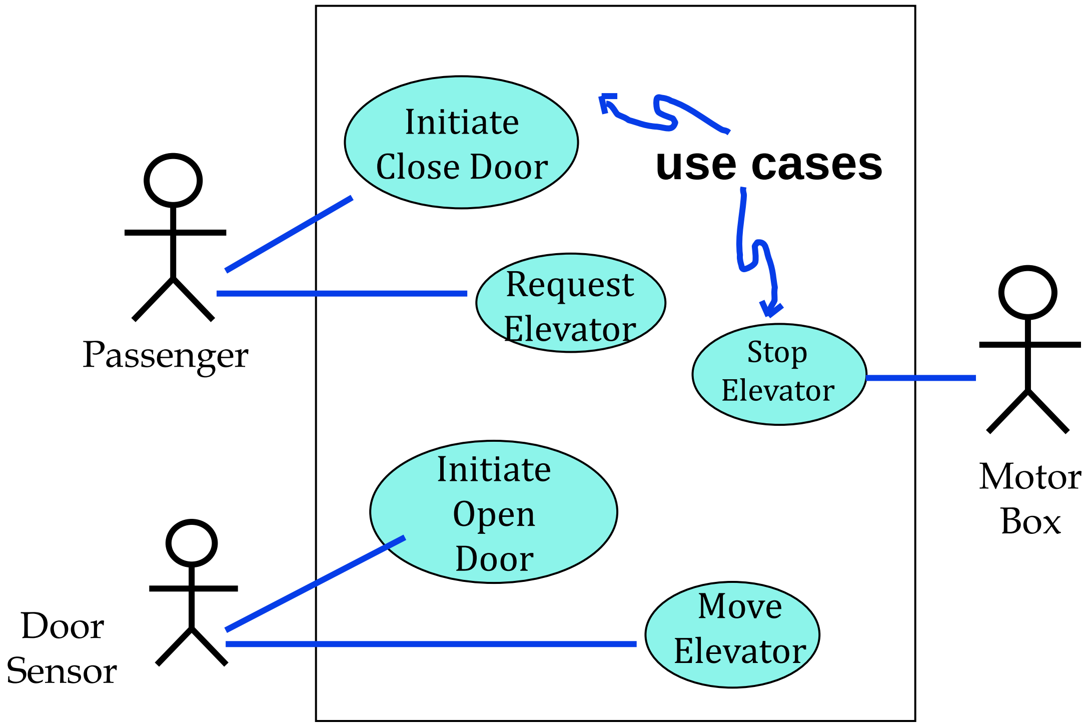
Ask what each one wants to do.



Interaction

Look for data and control inputs to the system.

Keep your mind on the *logical* intent of the interaction, not the specific *physical* flows.



Back-and-Forth

Generally, a use case involves “back-and-forth” across the system boundary.

<u>Order Elevator to Floor</u>

1. Close door
2. When door closed, accelerate
3. On approach to specified floor, slow down
4. On arrival at floor, open door

This suggests alternate paths:

- What if the door fails to close?
- What if a passenger requests an intervening floor?

Write the alternate paths as separate use cases.

Alternatives

Consider what can “go wrong.”

Build them as separate use cases.

<u>Order Elevator to Floor with Idiot in Door</u>

1. Close door
2. If blocked, reopen door
3. When completely reopened (and <3 tries, go to Step 1)
4. If 3 tries, make annoying beeping sound
5. ……

Workshop

Identify and name the use cases in the Case Study.

Draw a diagram, if you want.

9. Defining Use Cases

9

Defining Use Cases

To define a use case, follow this pattern (long form):

Actor: What role causes the use case to
execute?

Trigger: What causes the use case to execute

Pre-conditions: What must be true before the
use case can execute

Post-conditions: What must be true after the
use case has executed

Scenario: A description of just what happens

Requirements: A list of the requirements
addressed by the use case

Defining Use Cases

To define a use case, follow this pattern (short form):

~~Actor~~: Adds little

~~Trigger~~: Adds little

Pre-conditions: What must be true before the use case can execute

Post-conditions: What must be true after the use case has executed

Scenario: A description of just what happens

~~Requirements~~: Captured elsewhere

Trigger

A *trigger* is a data or control flow that initiates the use case.

Identify what causes the use case to execute.

Passenger orders elevator to floor

aka stimulus

Always write down the *logical meaning* of the trigger.

Pre- and Post-Conditions

Pre- and post-conditions say what must be true:

- before some use case, and
- after the use case is completed.

Elevator arrives at floor

Precondition:	Door closed and elevator stopped
Postcondition:	Door open

Idiot in Door

Precondition:	Door closing and obstruction detected
Postcondition:	Door opened

You may add a description.

Pre- and Post-Conditions

There may be several pairs of several conditions.

Request Elevator

Precondition:	Elevator at floor with door closed
Postcondition:	Door open initiated
	Request satisfied
Precondition:	No elevator at requested floor
Postcondition:	Request queued for arbitrary elevator

Pre- and Post-Conditions

Or you may split them.

Request Elevator with Elevator at Floor

Precondition:	Elevator at floor with door closed
Postcondition:	Door open initiated
	Request satisfied

Request Elevator with no Elevator at Floor

Precondition:	No elevator at requested floor
Postcondition:	Request queued for arbitrary elevator

Note that the state is now embedded within the name.

Scenario

A *scenario* is a list of steps describing the interaction between an actor and the system to effect some goal.

Write down the scenario in natural language.

Scenario: Passenger Enters Elevator

1. Create order for elevator to floor
2. Close door
3. Elevator moves
4. Elevator arrives at floor
5. Open door

Scenario: Passenger requests Elevator

1. Select elevator in requested direction
2. Create order for elevator to floor
3. Close door
4. Move elevator
5. Elevator arrives at floor
6. Open door

Scenario

A *scenario* is a list of steps describing the interaction between an actor and the system to effect some goal.

WARNING: Scenarios are procedural and can lead to brittle systems if implemented directly. They are used to elucidate the system's response ONLY.

Test Cases

Most test cases should take the form:

- establish pre-conditions
- inject stimulus to initiate
- verify actual post-conditions against expected
- issue a pass/fail indication

Hmmm….

Where would I find these?

Workshop

Write pre- and post-conditions for the following use cases:

- 1: Simple Workout
- 2: Multi-Lap Workout
- 3: Achieve Pace Over Distance Goal

10. Factoring Use Cases

10

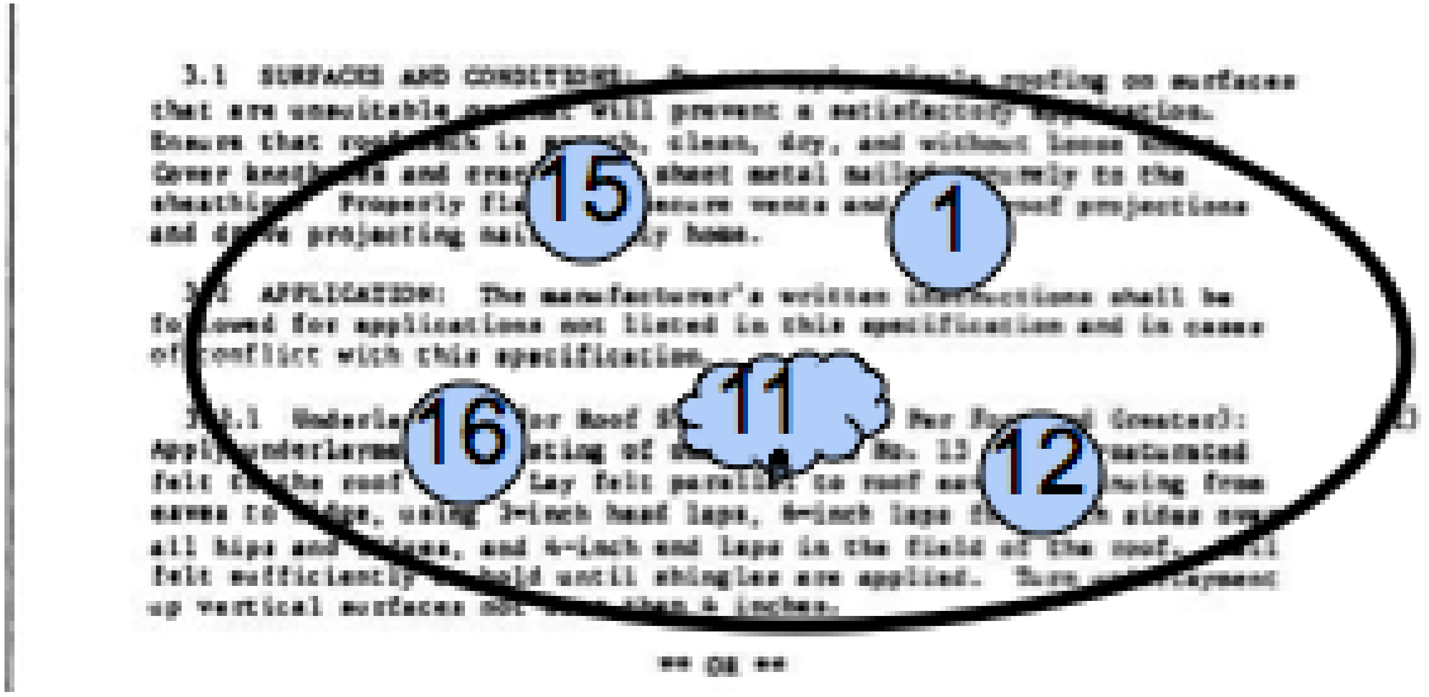
The Whole

Now that we have the use cases, it pays to look at them as a whole to see if we can find:

- inconsistencies in terms
- inconsistencies in abstraction level
- duplication of requirements
- elements that can be factored out

Size Matters

A single use case use case may encompass several of the original candidate requirements.



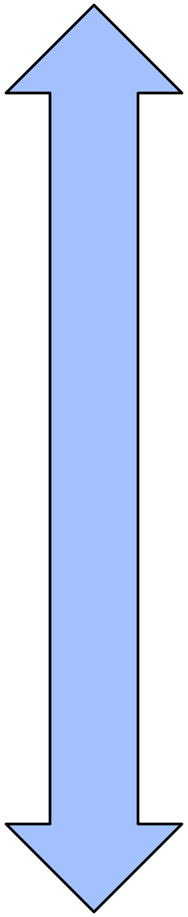
3.1 SURFACES AND CONDITIONS: [illegible] roofing on surfaces that are unsuitable [illegible] will prevent a satisfactory [illegible]ation. Ensure that roof[illegible] is [illegible], clean, dry, and without loose [illegible] Cover knot[illegible] and crac[illegible] sheet metal nailed securely to the sheathing. Properly fla[illegible] secure vents and [illegible] roof projections and d[illegible] projecting nai[illegible]y home.

3.2 APPLICATION: The manufacturer's written instructions shall be followed for applications not listed in this specification and in cases of conflict with this specification.

3.2.1 Underla[illegible]or Roof S[illegible] Per Foot and Greater): Apply underlayment [illegible]ting of [illegible] No. 15 [illegible]saturated felt [illegible] the roof [illegible] lay felt parallel to roof [illegible]ning from eaves to [illegible]ge, using 3-inch head laps, 6-inch laps [illegible] sides over all hips and [illegible], and 4-inch end laps in the field of the roof. [illegible] felt sufficiently [illegible] hold until shingles are applied. Turn [illegible]ayment up vertical surfaces not [illegible] than 4 inches.

** 08 **

Use Case Diagram



Use cases can be:

- Very big, involving multiple elements
- Big, involving a single, complex interaction
- Single requirement
- Very small (and very useless)

want to be
We ~~are~~ here

Normalize Nouns

Ensure that the terms used in the use cases are either:

- exactly those you have defined, or
- can be replaced by those you have defined, or
- you provide a definition

Correlate them to existing requirements for

- identification
- completeness

But do not rewrite them.

Pace	Workout
Lap	Log
Multi-lap	Button
Device	Heart-rate
Track	…..

Normalize Verbs

Ensure that the verbs used in the use case use cases are:

- limited (i.e. there are but a few of them), and
- clear, and if special to the subject matter…

… correlate them to existing requirements for

- identification
- completeness

But do not rewrite them.

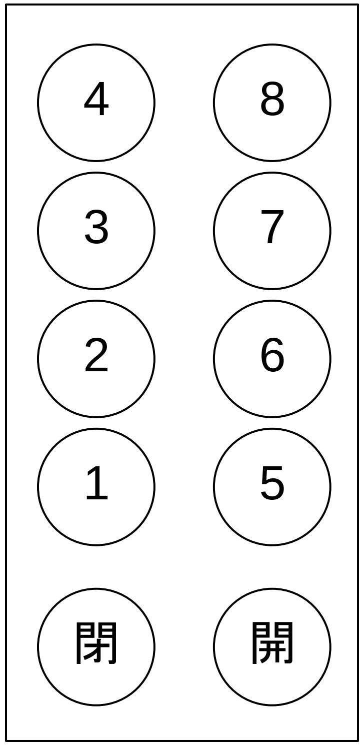
Add	Display
Delete	Alert
Read	Clear
Find	Log
Show	…..

Inconsistent Abstraction Level

Check that terms used in the use cases are at the same abstraction level.

Passenger pushes button.

Passenger orders elevator to floor.

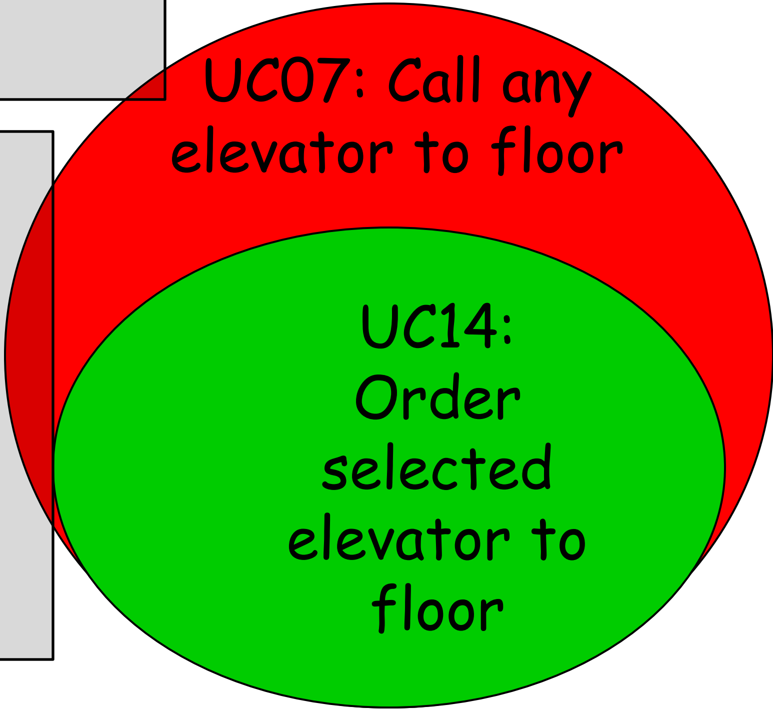


Factor Out Common Elements

When you find common elements, factor them out.

<u>Scenario UC07: Passenger Enters Elevator</u>
Create order for elevator to floor
Close door
Elevator moves
Elevator arrives at floor
Open door

<u>Scenario UC14: Passenger requests Elevator</u>
Select elevator in requested direction
Create order for elevator to floor
Close door
Move elevator
Elevator arrives at floor
Open door


UC07: Call any elevator to floor
UC14: Order selected elevator to floor

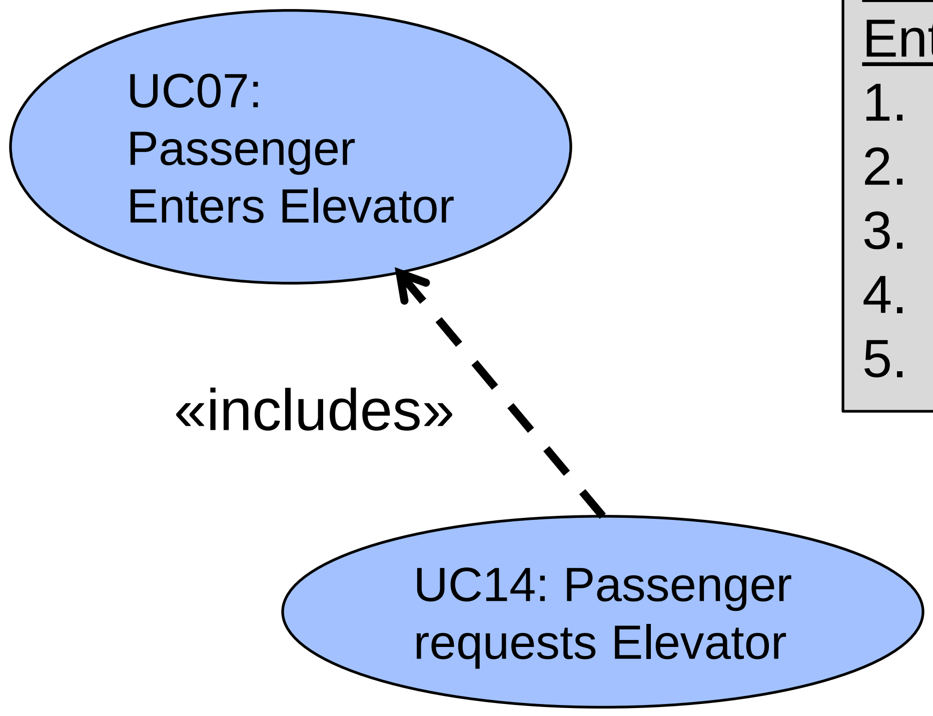
Additional Constructs

Other relationships exist within the system boundary.

They are:

- «includes»
- «extends»
- and generalizes (*deprecated*)

Includes


UC07: Passenger Enters Elevator
«includes»
UC14: Passenger requests Elevator

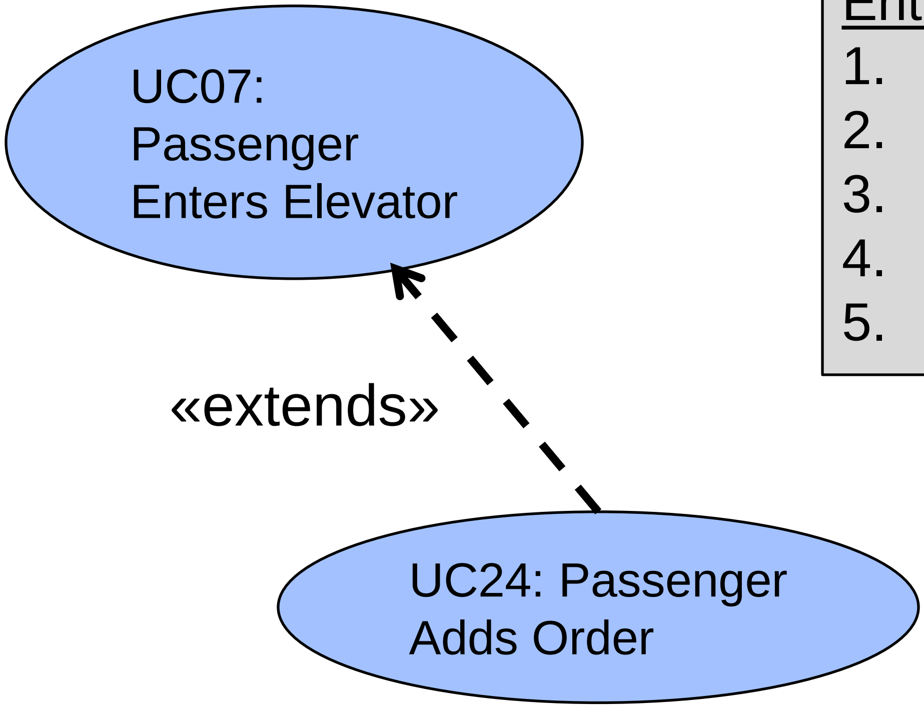
<u>Scenario UC07: Passenger Enters Elevator</u>

1. Create order for elevator to floor
2. Close door
3. Elevator moves
4. Elevator arrives at floor
5. Open door

<u>Scenario UC14: Passenger requests Elevator</u>

1. Select elevator in requested direction
2. Include UC07

Extends



<u>Scenario UC07: Passenger Enters Elevator</u>

1. Create order for elevator to floor
2. Close door
3. Elevator moves
4. Elevator arrives at floor
5. Open door

<u>Scenario UC24: Passenger Adds Order</u>

Duplicates UC07 Steps 1~3
Create additional order, if not past floor
Repeat UC07 Steps 2~5 for each order

One More Time…

Each requirement is	if not
Identified	Identify it
Unique	Reconcile it with duplicates
Coherent	Clean up the writing
Unambiguous	Disambiguate
Testable	Revisit the pre/postconditions

Workshop

Reconstruct the existing use cases and verify the terms are consistent.

11. Information Gathering

11

Gathering Information

What do you do if you don’t have enough information?

You talk to the technical experts:

- Limit meetings to one hour
- Check *constantly*
- Write down what you learned
- Ask the expert to check your understanding
- Do it again

So each robot has one arm that can move in two dimensions, not three? Right?

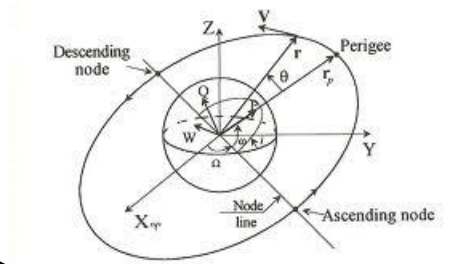
Technical Notes

Writing is *not* “extra.” It forces you to organize your thoughts.

Technical notes capture any topic of interest.

- Informal
- Short
- Common understanding
- Always incorporated into the executable models somehow

Orbital Mechanics for Dummies



They will also avoid having the same interview again.

And again…. and again…. and again…. and again…. and ...

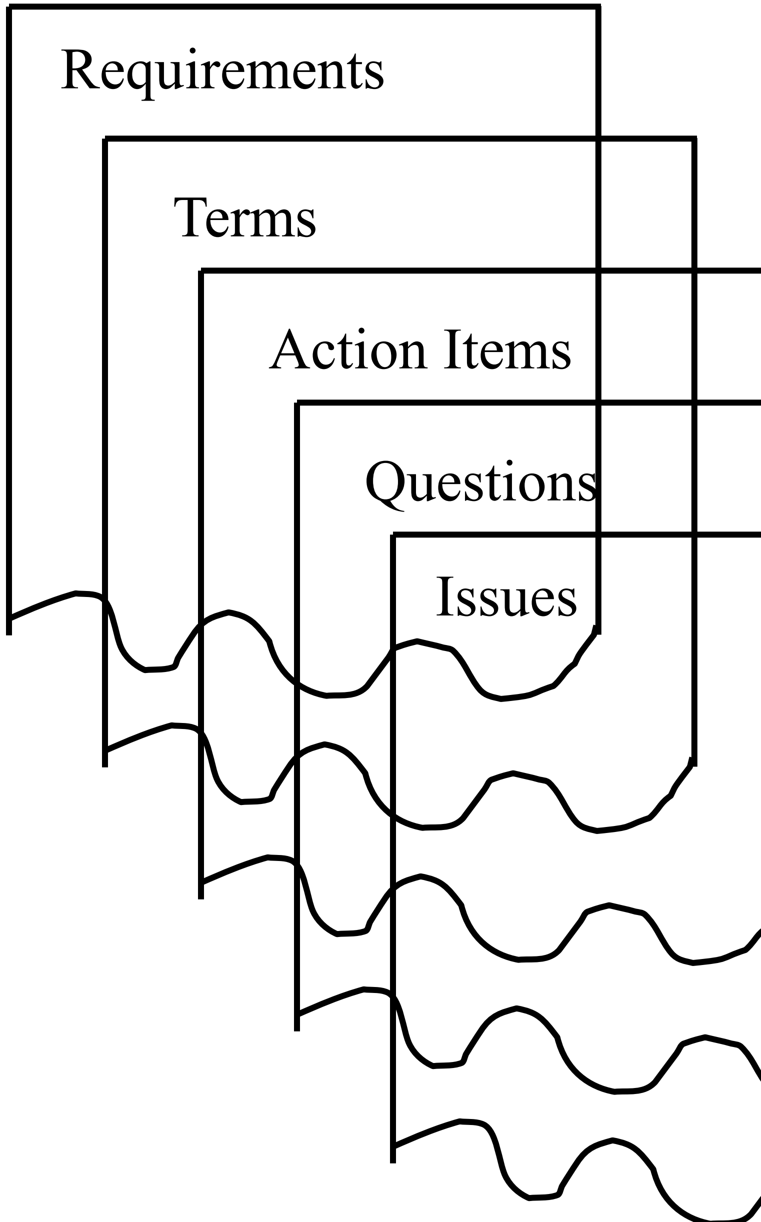
As You Write …

Note down additional:

- Requirements
- Terms

- Action Items
- Questions for experts
- Issues

These are “living documents.”

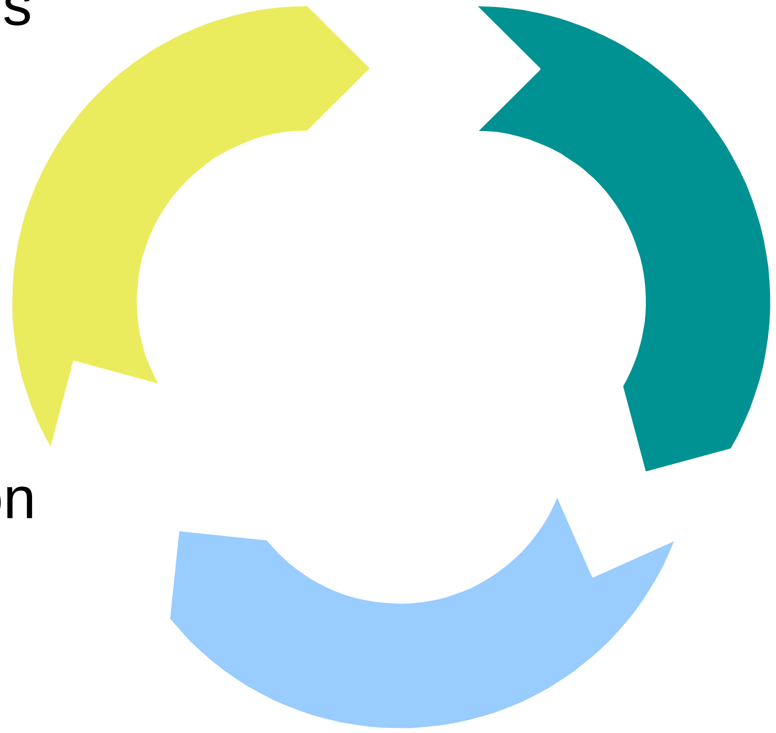


Intermediate Review

The systems engineers/experts should review your:

- technical notes
- definitions of mysterious terms
- factoring requirements

and answer your questions
as you proceed.



The cost of using faulty information is high and the cost increases exponentially the longer you continue to use faulty information!

12 Activity Diagram

12

Activity Diagrams

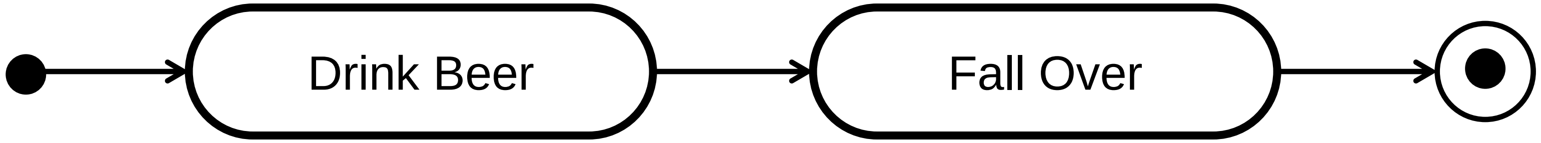
The purpose of building an activity diagram is to

- capture shared understanding of sequencing and processing,
- and make obvious the opportunity for concurrency.

Basics of Activities

Activity diagrams can show:

- Activities
- Transitions
- Initial node
- Final node

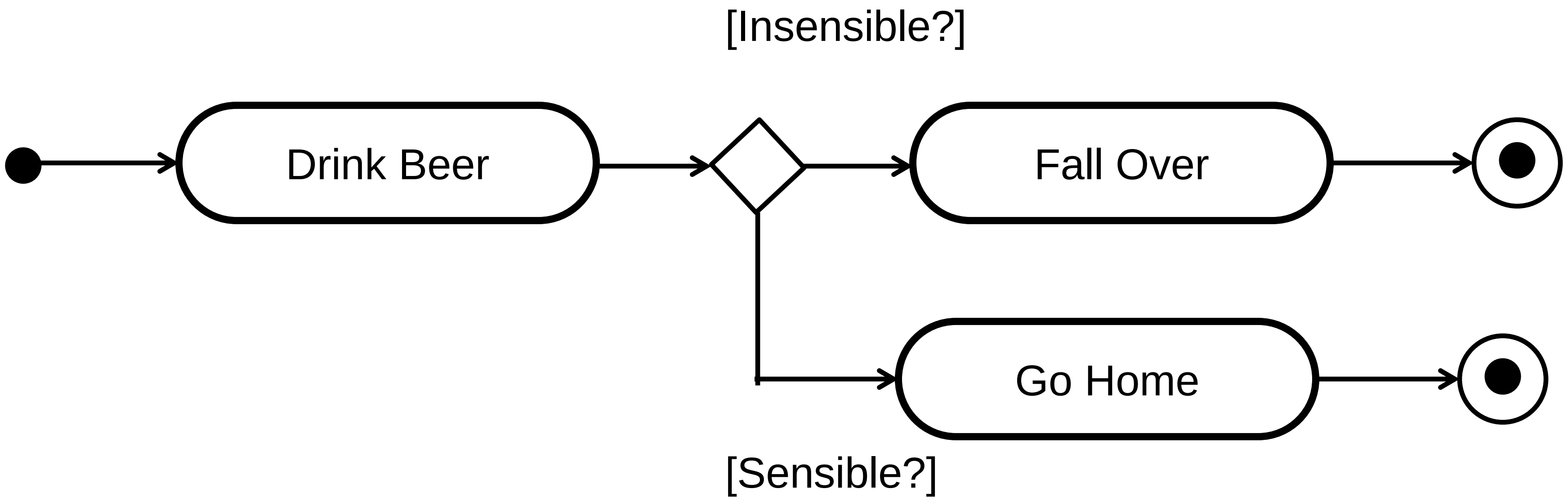


The transitions indicate the sequencing.

Decisions

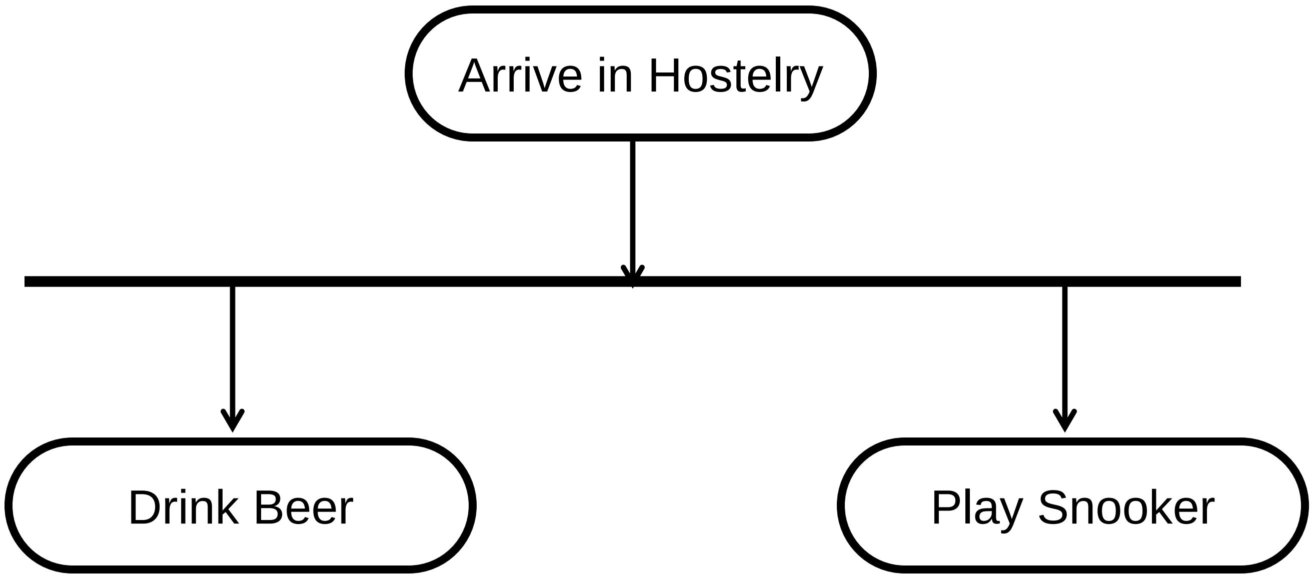
Activity diagrams may also make decisions.

- Decision node (diamond) shows decision
- [guards] indicate conditions that must be true
- Decision node also used for merge



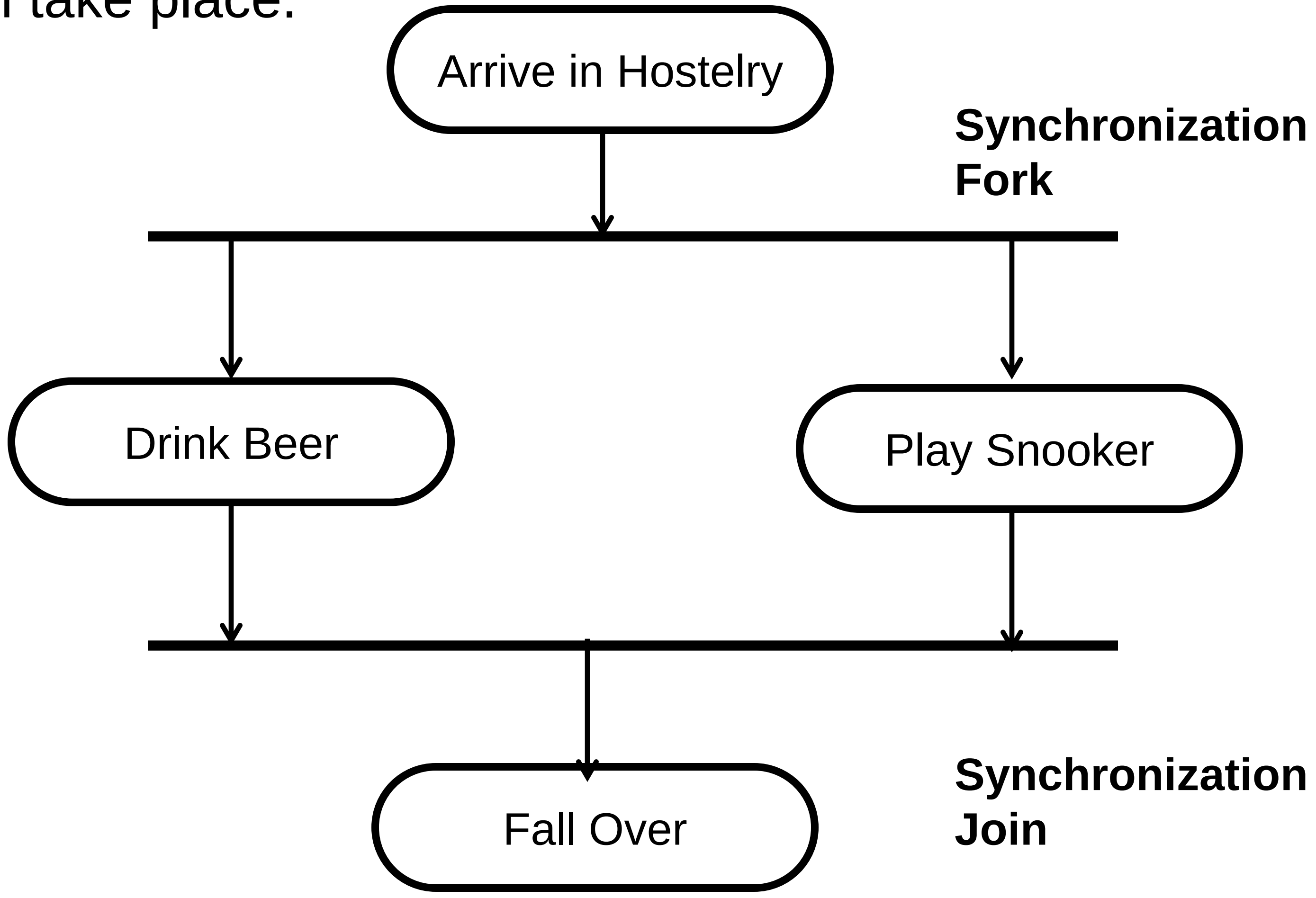
Parallel Activities

Several actions make take place in parallel.



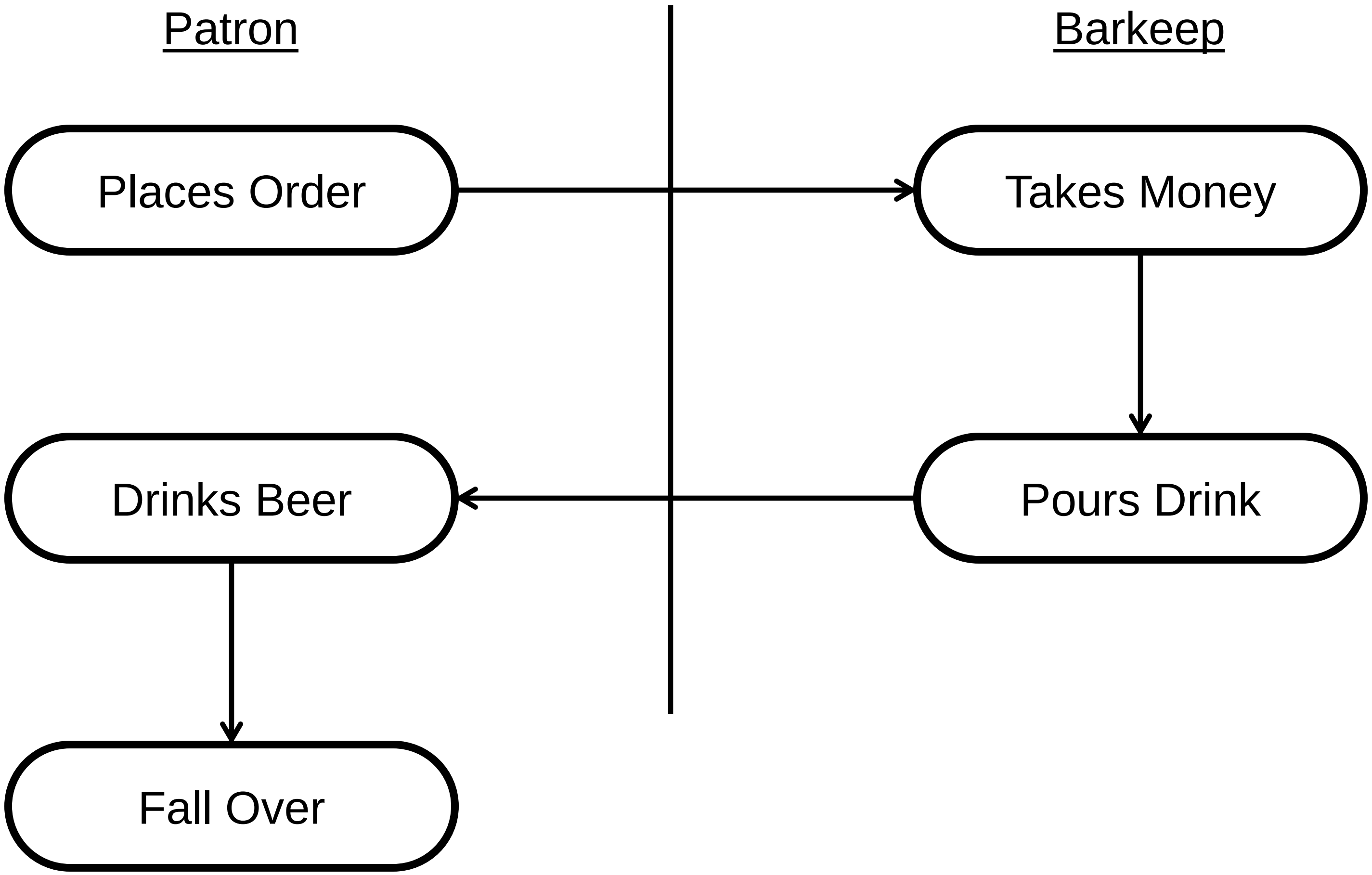
Parallel Activities

Parallel activities may need to terminate before another activity can take place.



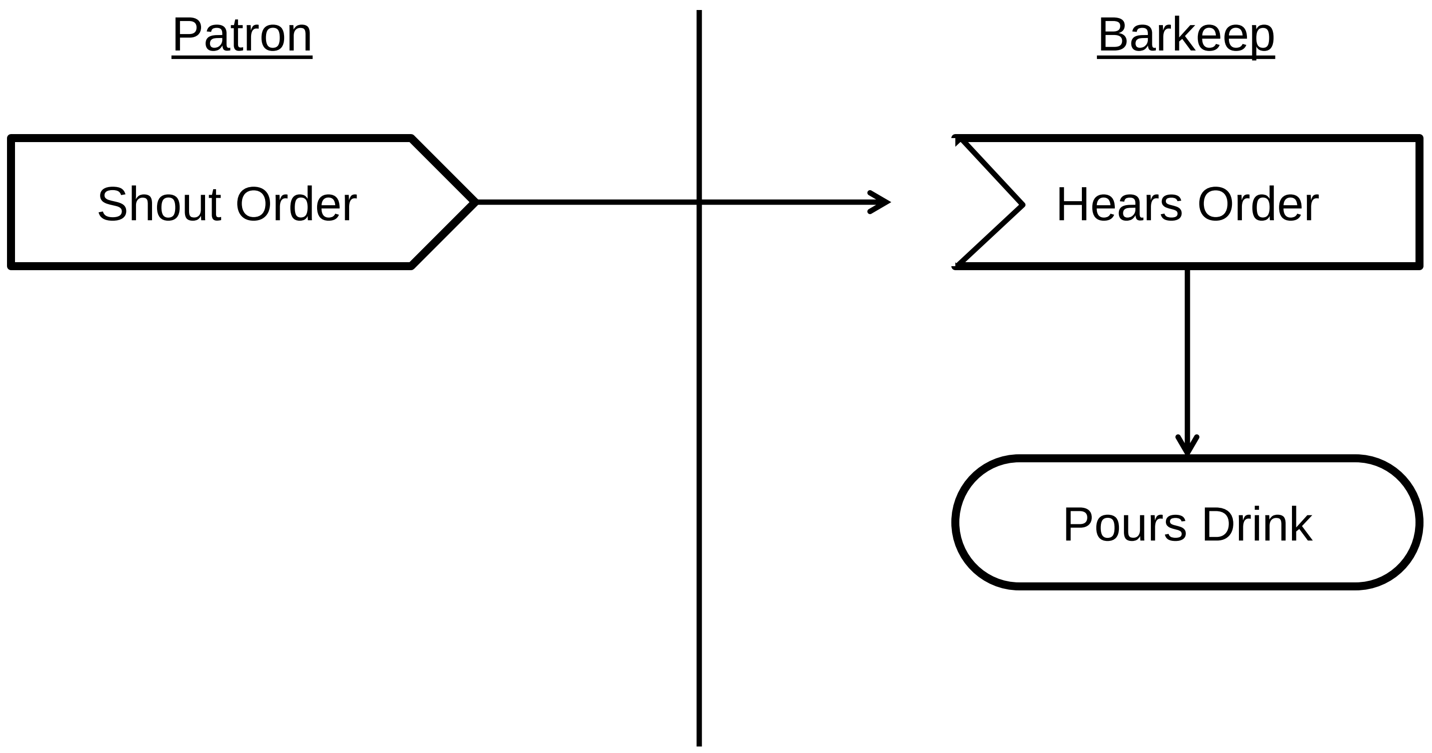
Swimlanes

To clarify actors' roles, activities are aligned inside swimlanes.

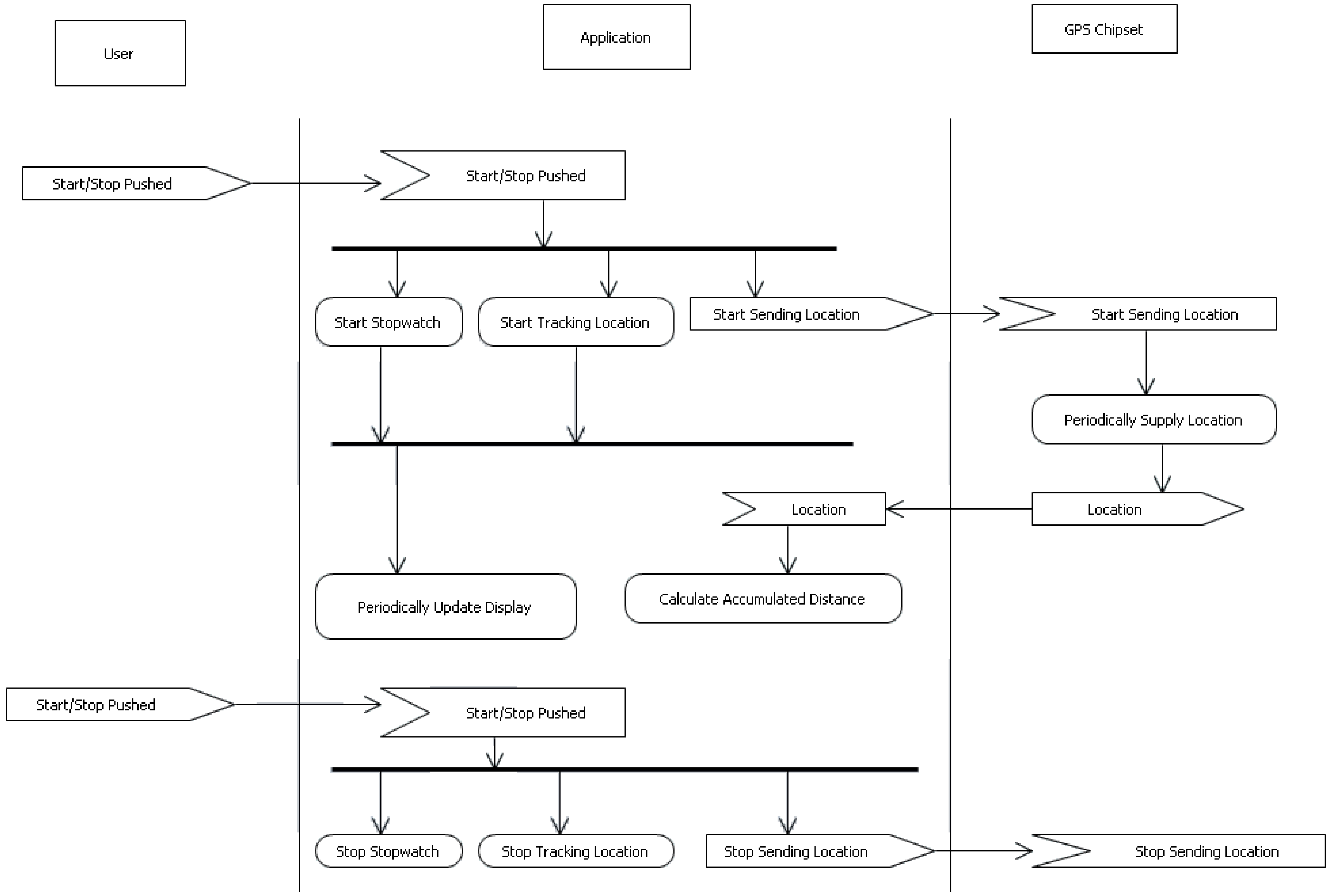


Signals

Actions may send signals and accept the corresponding event.



Activity Diagram



Workshop

Draw an activity diagram for use case #5.

13. Sequence Diagrams

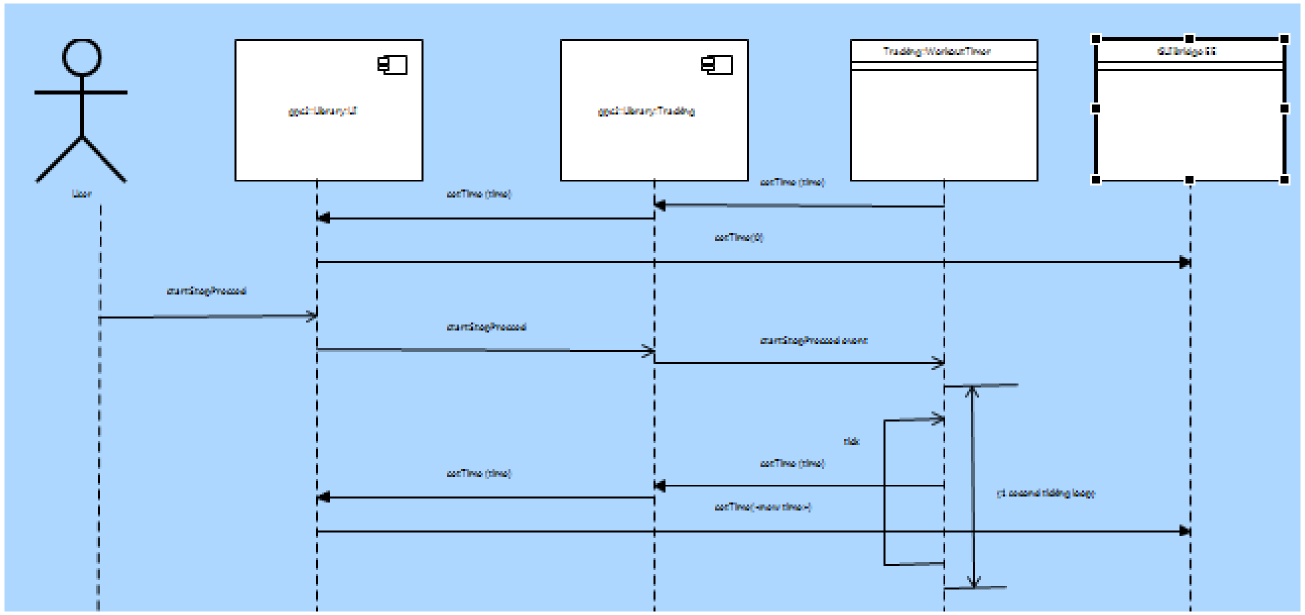
13

(Message) Sequence Diagrams

A sequence diagram shows how processes operate with one another and in what order.

Among other elements, a sequence diagram has:

- Lifelines
- Messages
- Timing



Lifelines

Anything that has its own behavior, that can occur concurrent with others, can be said to have a *lifeline.*

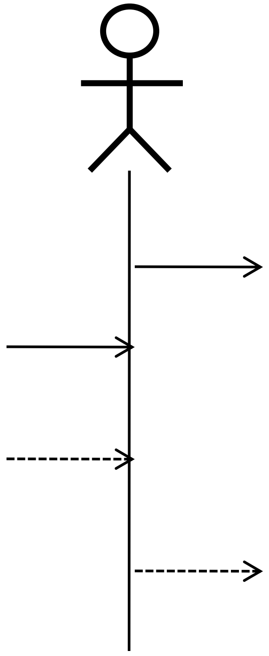
The lifeline shows how its owner behaves over time.

A lifeline can be a:

- Actor
- Component
- Instance
- External Entity
- Class

These are informal, though they may be connected to formal things later.

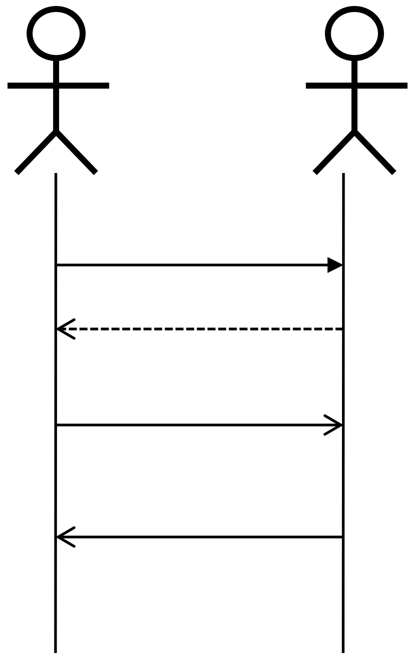
It can send messages to others.



Messages

Messages can be

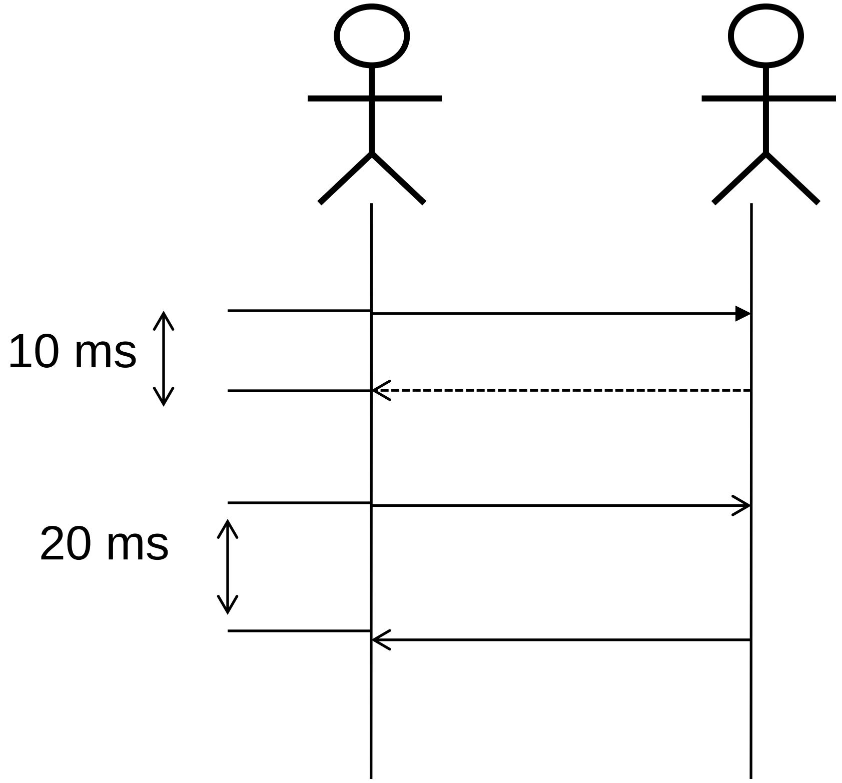
- Synchronous (wait for return ➡)
- Return ($\dashleftarrow$)
- Asynchronous (send and forget $\longrightarrow$)



Timing

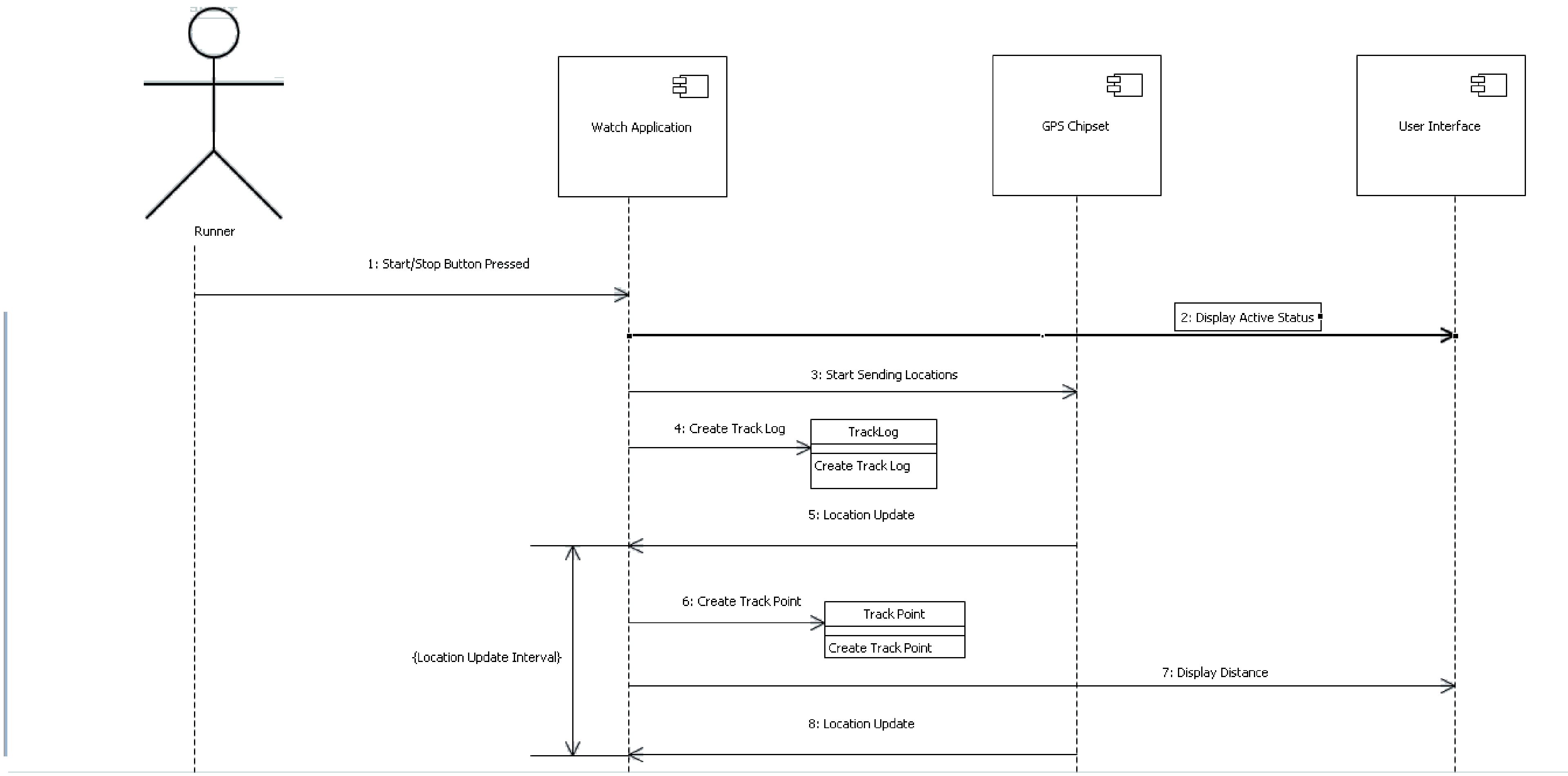
Timing elements include:

- Marks and
- Spans



Sequence Diagrams

Build a sequence diagram if it helps detail your understanding.



14. Packaging the Materials

14

Organizing the Elements

The result of all this work is mostly

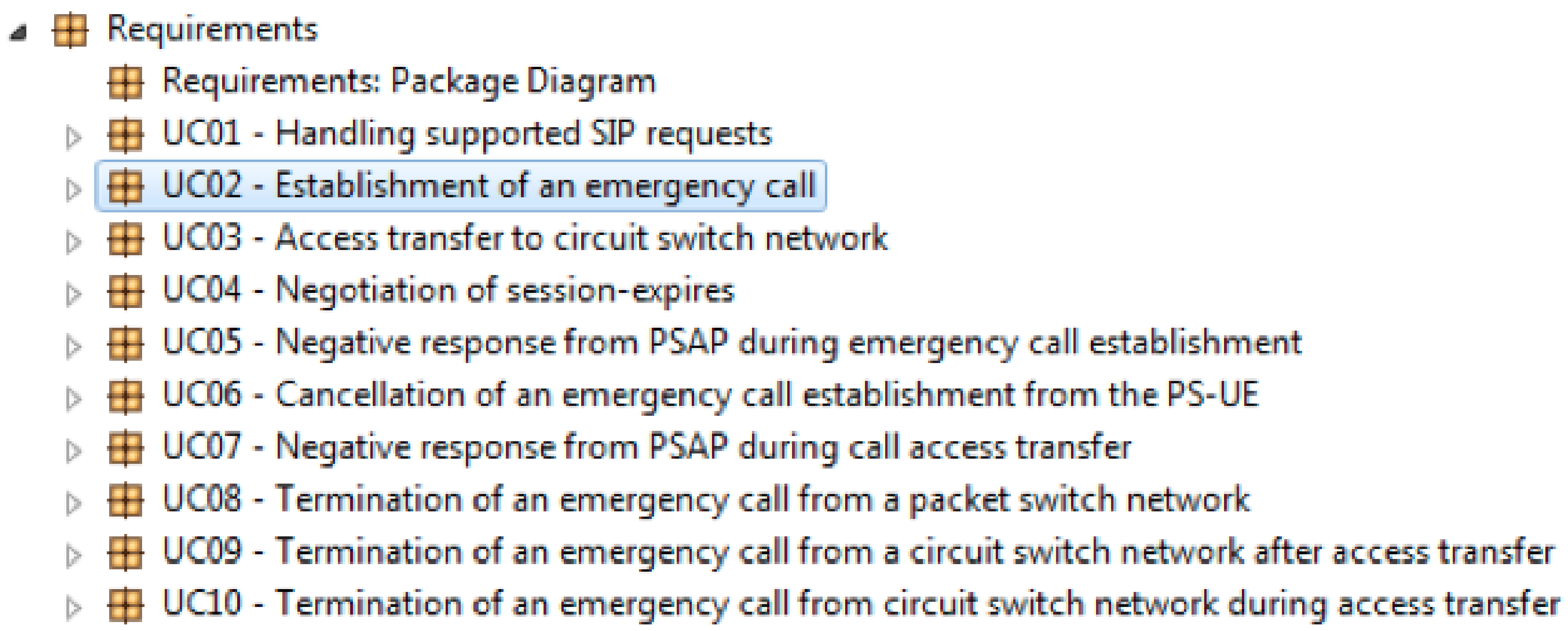
UNDERSTANDING

but the work must be packaged up for

- review and
- ease of access.

Requirements

We create a package named “Requirements”, with the use cases listed underneath.

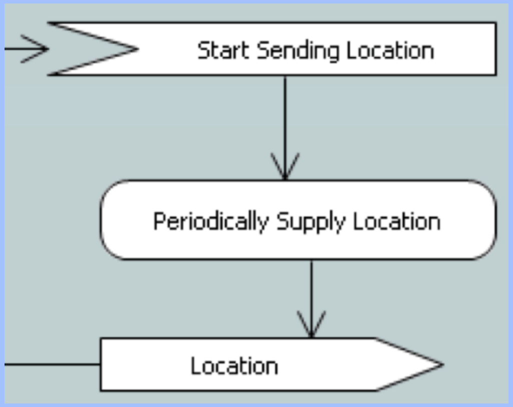
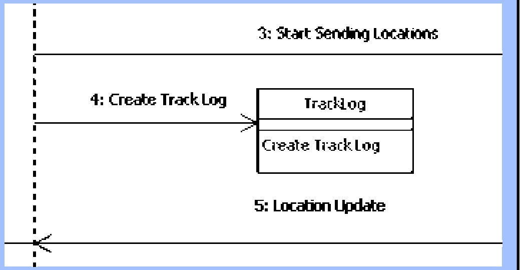


Requirements
- Requirements: Package Diagram
- UC01 - Handling supported SIP requests
- UC02 - Establishment of an emergency call
- UC03 - Access transfer to circuit switch network
- UC04 - Negotiation of session-expires
- UC05 - Negative response from PSAP during emergency call establishment
- UC06 - Cancellation of an emergency call establishment from the PS-UE
- UC07 - Negative response from PSAP during call access transfer
- UC08 - Termination of an emergency call from a packet switch network
- UC09 - Termination of an emergency call from a circuit switch network after access transfer
- UC10 - Termination of an emergency call from circuit switch network during access transfer

Use Cases

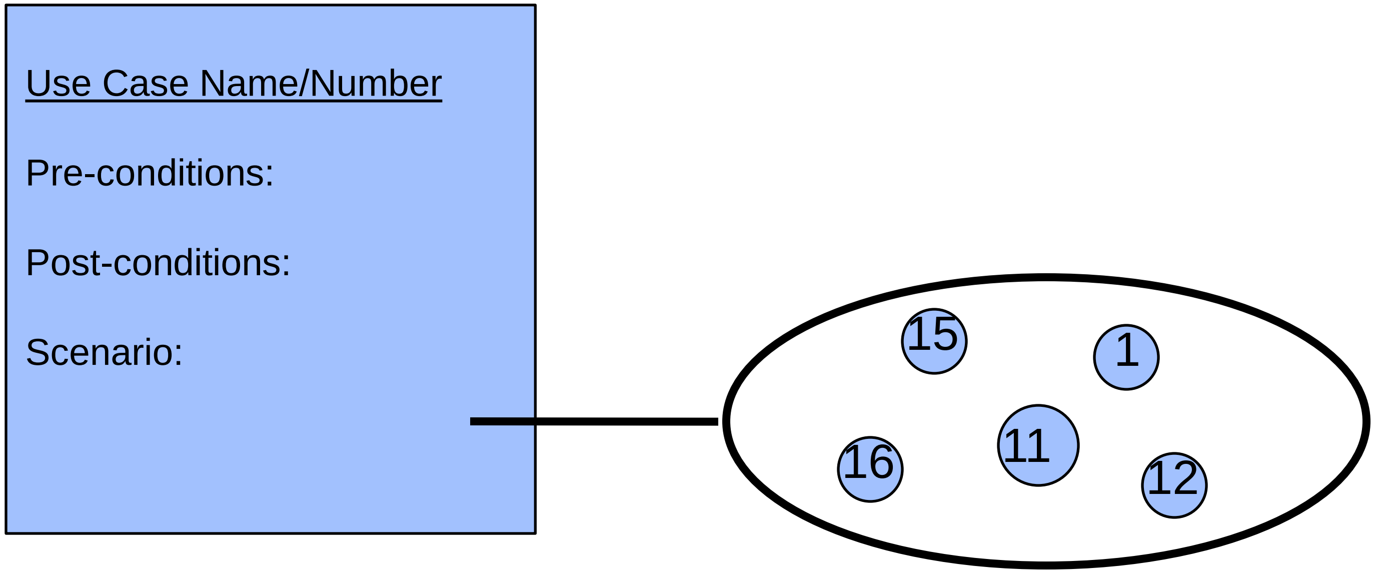
Each use case shall contain:

- a description
- an activity diagram, and optionally
- a sequence diagram

<u>Use Case Name/Number</u> Pre-conditions: Post-conditions: Scenario:	 Start Sending Location Periodically Supply Location Location	 3: Start Sending Locations 4: Create Track Log TrackLog Create Track Log 5: Location Update

Use Case Description

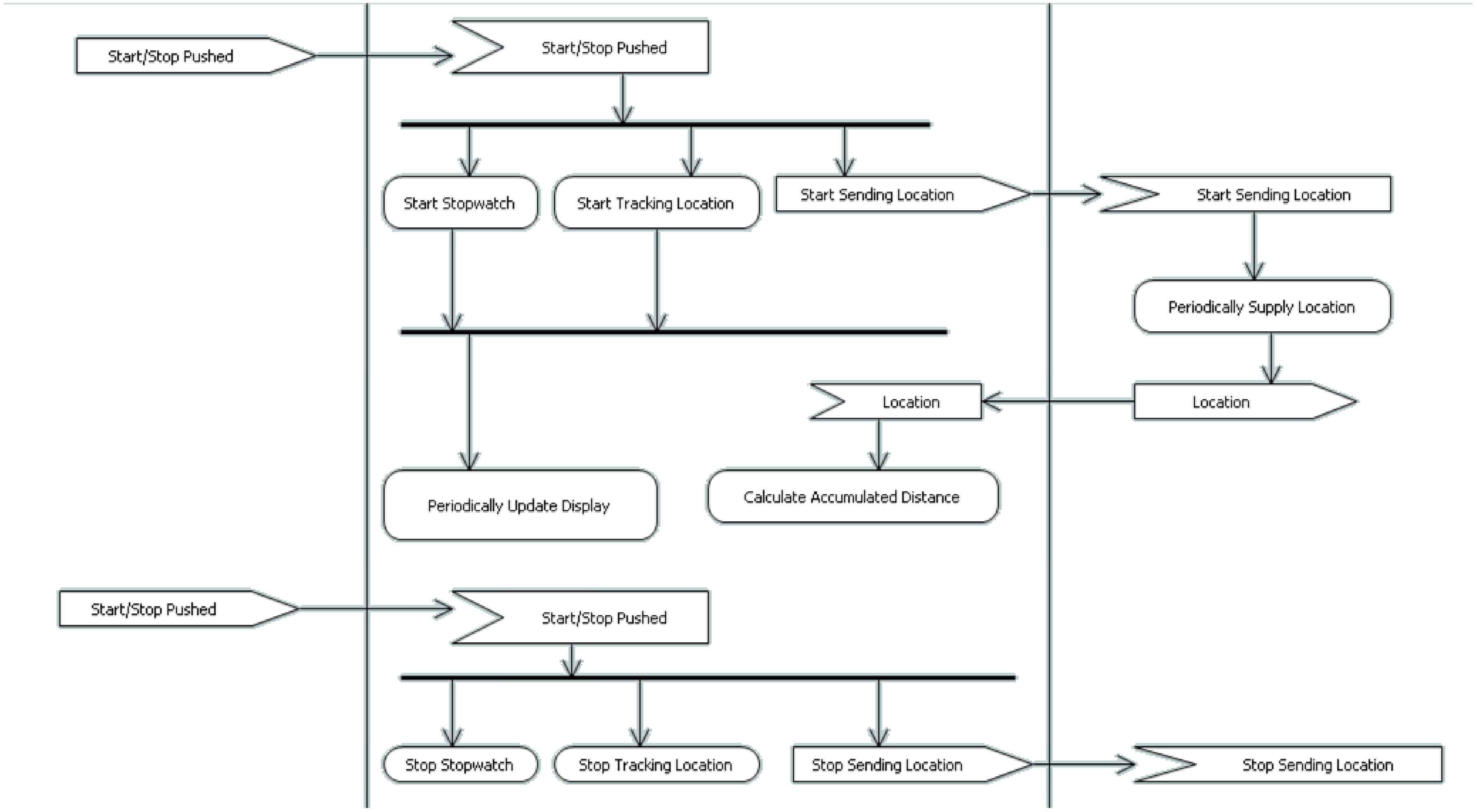
The use case description shall contain the description



and *could* cross-reference to the requirements it implements.

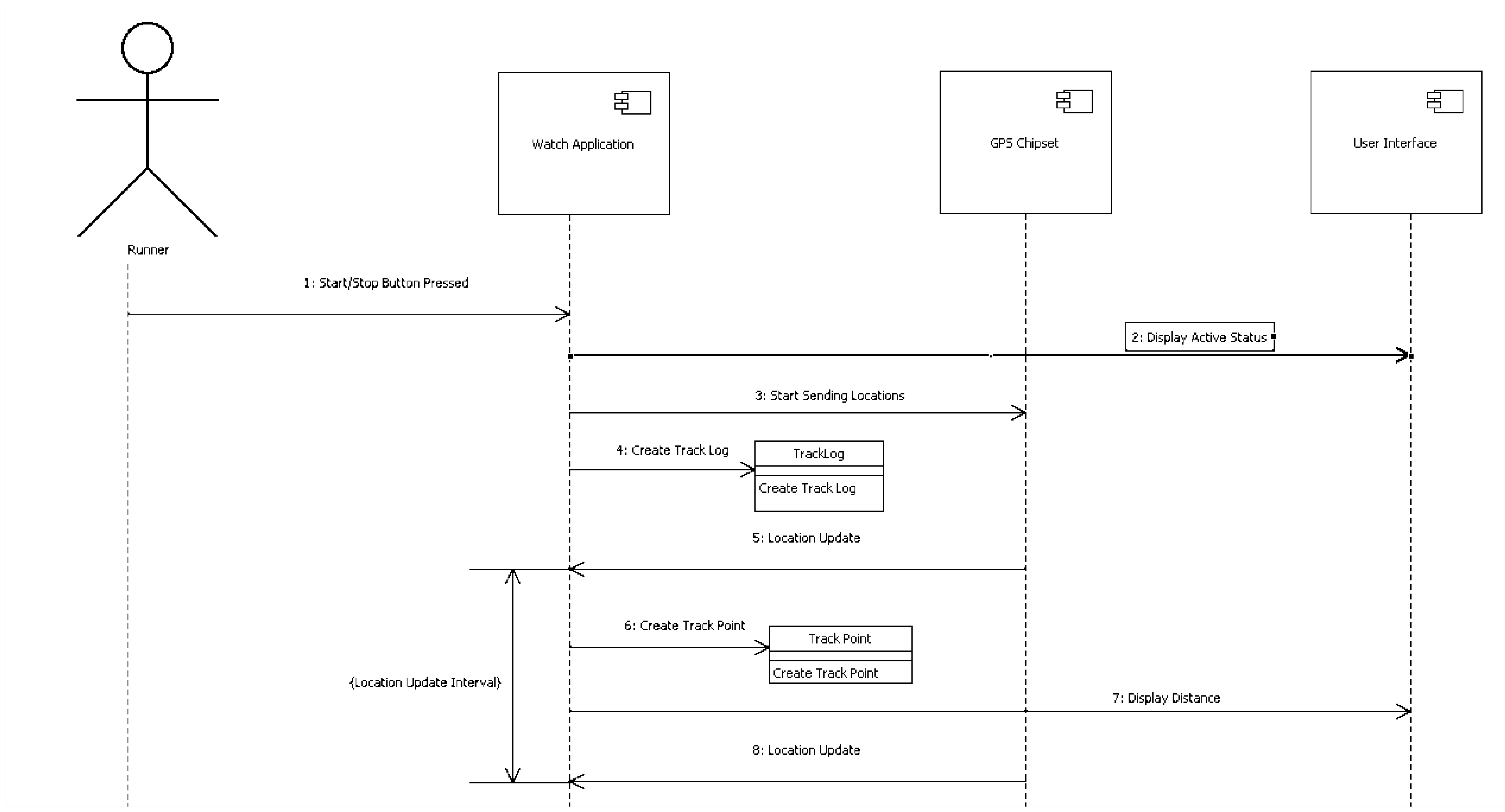
Activity Diagram

The Activity Diagram captures the sequencing and processing.



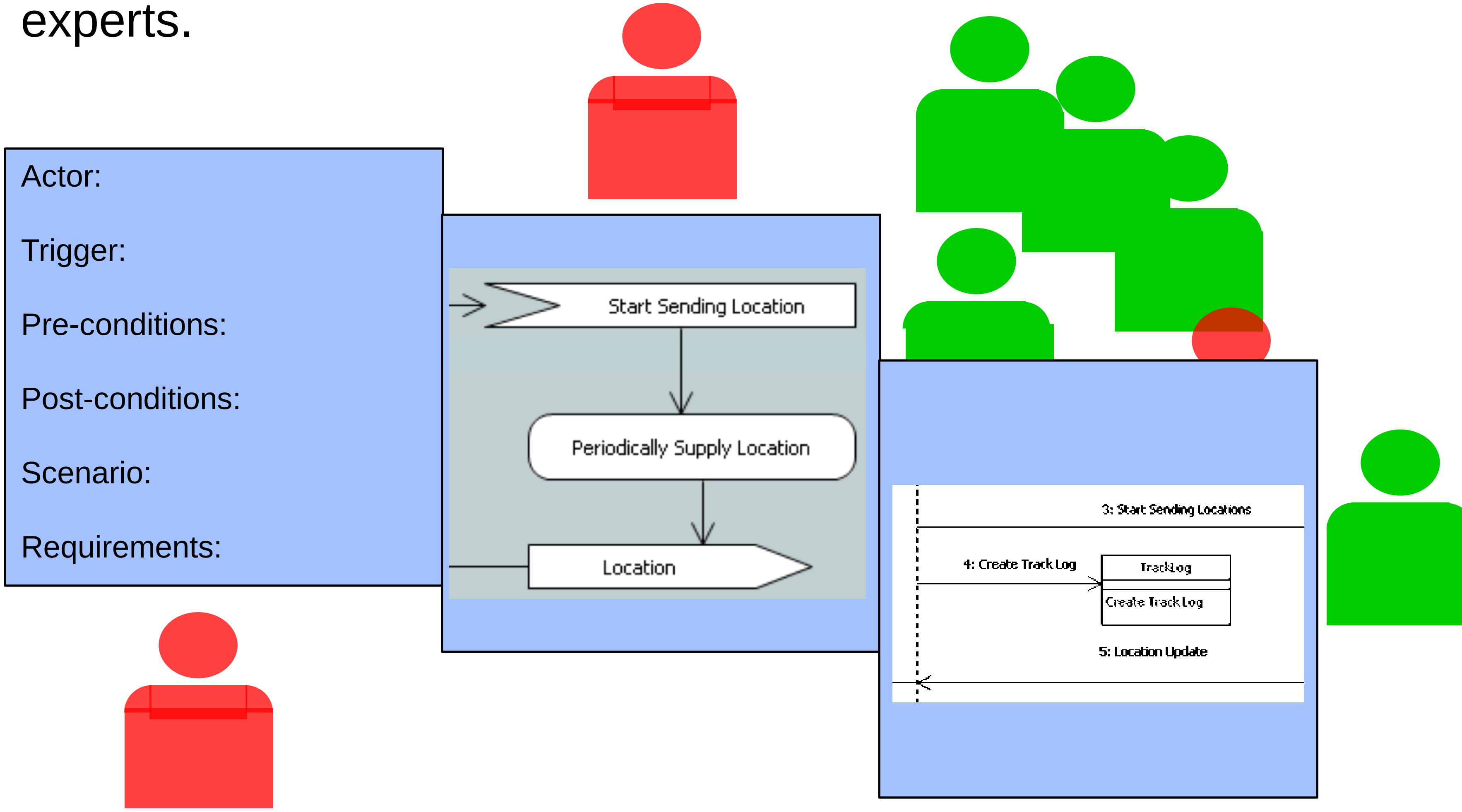
Sequence Diagram

And the Sequence Diagram captures detailed message flow.



Review

All of this needs to be reviewed by the customers and their experts.



Actor:

Trigger:

Pre-conditions:

Post-conditions:

Scenario:

Requirements:

Also Keep

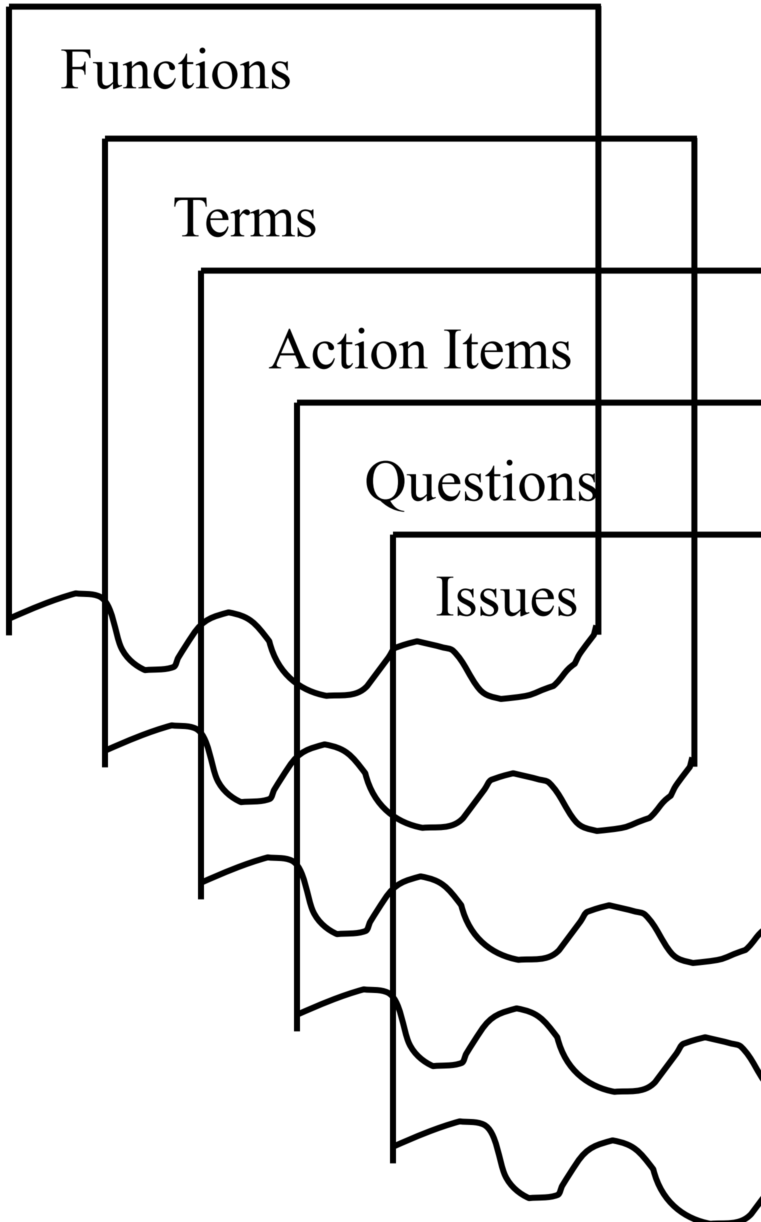
Keep the descriptions of the:

- Requirements, and
- Terms

You'll need them for the next stage.

- Action Items
- Questions for experts
- Issues

should be empty!



C: Wrap Up

15. What We Did

15

Levels of Commitment

Consequently, we must commit incrementally.

- Natural language and informal diagrams
  - Use cases
  - Activity diagrams
  - Sequence diagrams
- Structural models
  - Components & Interfaces
  - Class models
  - Data types
- Behavioral models
  - State models
  - Activities

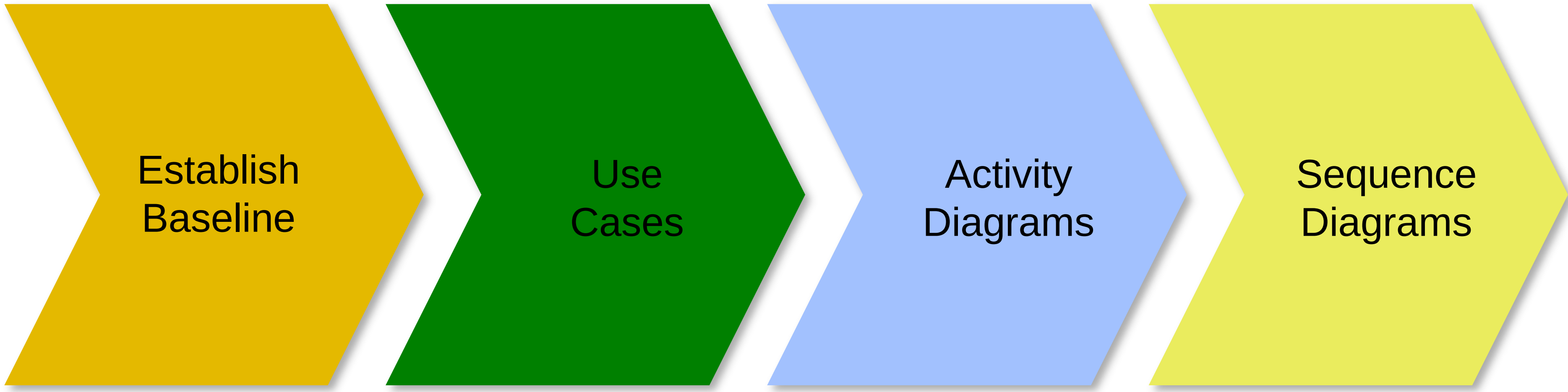
This course

Requirements Clarification Process

The process is:

- Find all your people, resources, practices, etc.
- Find out what the system-as-a-whole does
- Determine the precise behavior of each use case
- And establish how it communicates with others

But it's really all about learning about the problem.



Requirements Clarification Process

We'll show you how to:

- determine what you have, and
- how well it meets your needs
- gather information to build executable models
- investigate questionable use cases
- organize information ready to build executable models

It's a bootstrapped process

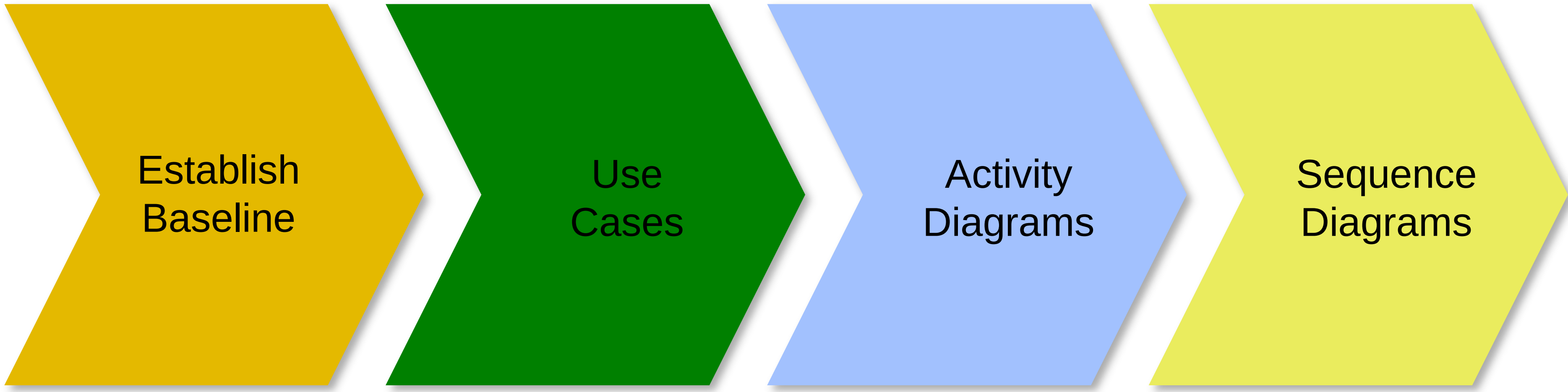


Table of Contents

16. What's Next

16

Abstraction

Now that we have approved use cases, and it’s all in our heads, it time to:

THINK

From that thinking, we create abstractions.

Build the Models

The following step is the build the models, and commit ourselves further.

Levels of Commitment

- Natural language and informal diagrams
  - Use cases
  - Activity diagrams
  - Sequence diagrams
- Structural models
  - Components & Interfaces
  - Class models
  - Data types
- Behavioral models
  - State models
  - Activities

Next course

THE END