Scenarios

Unlikely scenario

?

Scenario Analysis

- From Scenarios to Use Cases
- Scientific cycle of inquiry
- London Ambulance Dispatch System

- Goals and Scenarios
- Misuse Cases

Scenarios: Abstract -> Concrete

Example isn't another way to teach, it is the only way to teach. [Albert Einstein]

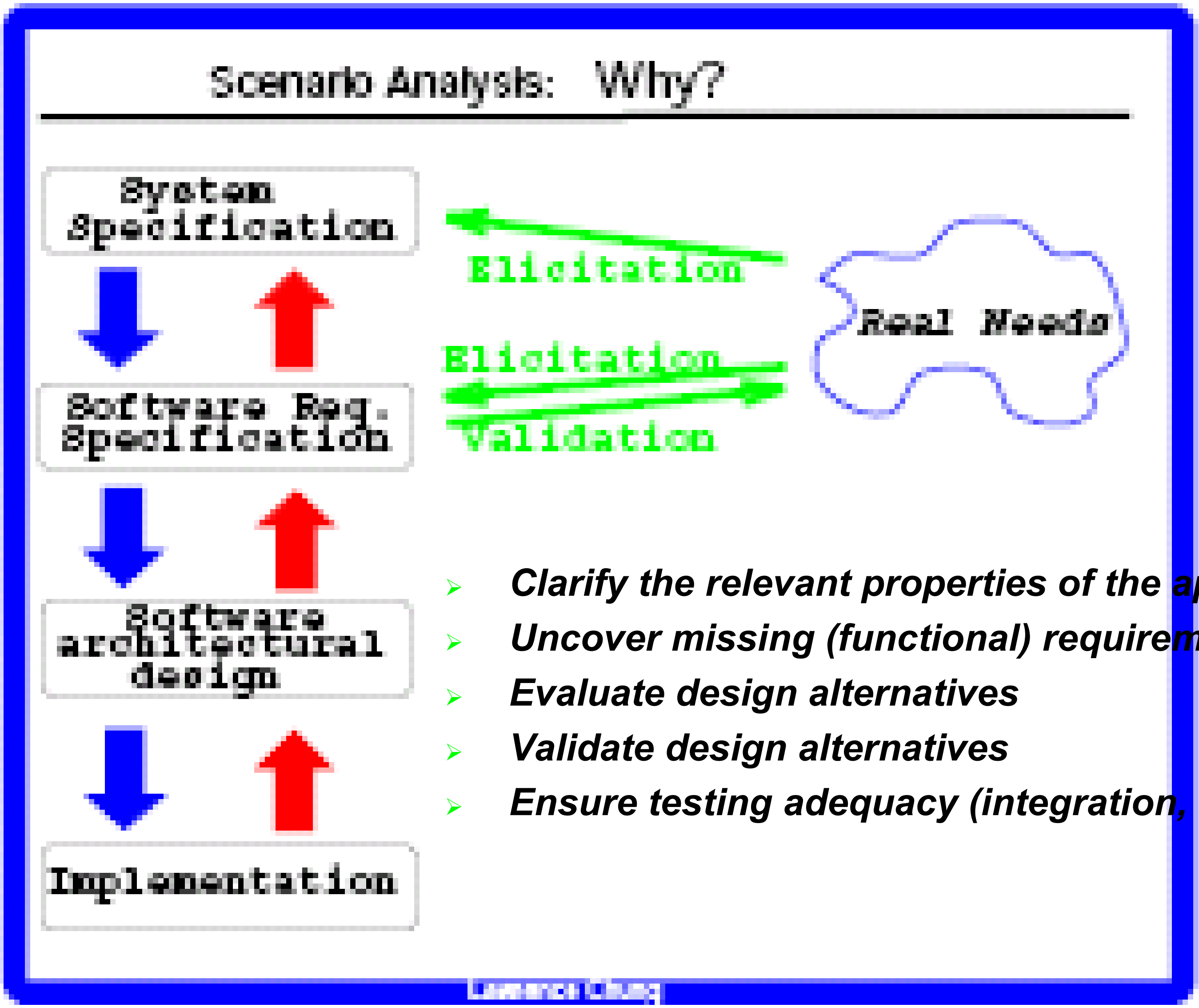
Example is not the main thing in influencing others, it is the only thing. [Schweitzer]

Scenarios

- From Wikipedia
  - a synthetic description of an event or series of actions and events
  - an account or synopsis of a projected course of action, events or situations

- From Software Engineering/Human Computer Interaction
  - Scenarios describe how users of the system interact with the system to achieve their particular tasks
  - Scenarios usually refer to representative instances of user-system interactions.

- From Collaborative Computing Perspective:
  - Scenarios describe how users interact with one another, using hardware and software, to achieve their goals

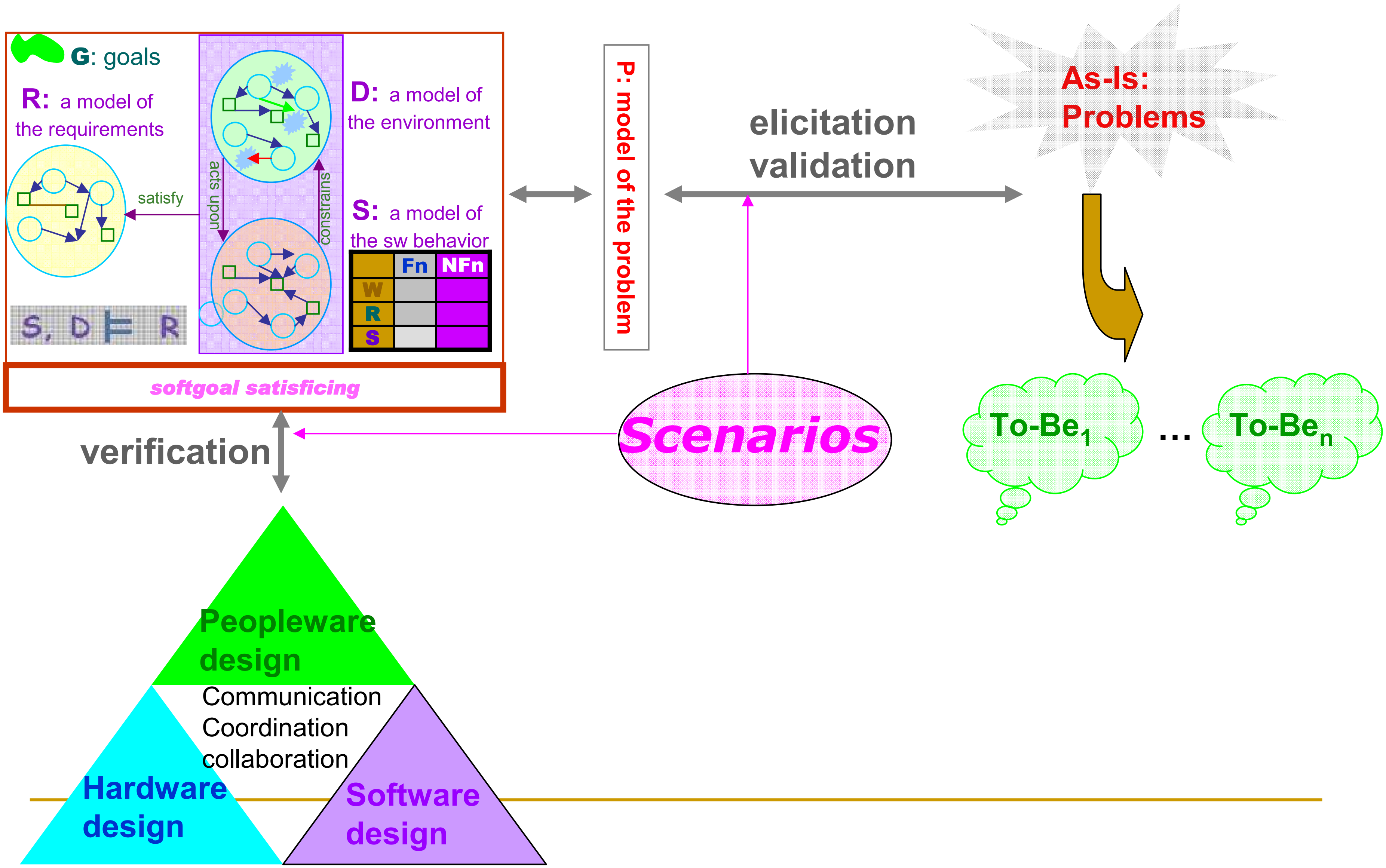


Scenario Analysis: Why?

- *Clarify the relevant properties of the application domain*
- *Uncover missing (functional) requirements.*
- *Evaluate design alternatives*
- *Validate design alternatives*
- *Ensure testing adequacy (integration, validation, ...)*

$M^G, Prog^G \models S^G; S^G, D^G \models R^G; R^G, D^G \models G; (G \models \neg P) \vee (G \mid\!\sim \neg P)$

Scenario Analysis



From Scenarios to Use Cases

Types of Scenarios [from Brugge's lecture notes]

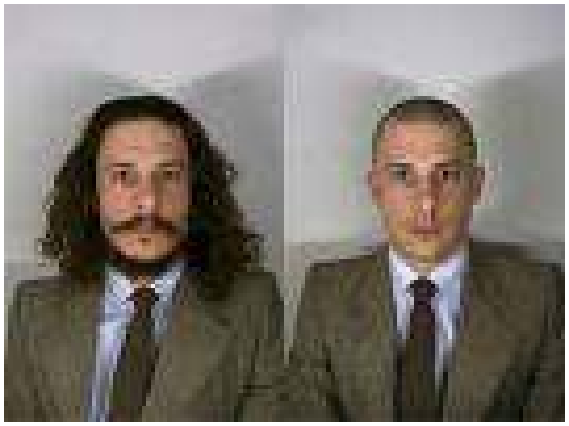
- **As-is scenario**:
  - Used in describing a current situation. Usually used in re-engineering projects. The user describes the system.
- **Visionary scenario**:
  - Used to describe a future system. Usually used in greenfield engineering and reengineering projects.
  - Can often not be done by the user or developer alone
- **Evaluation scenario**:
  - User tasks against which the system is to be evaluated.
- **Training scenario**:
  - Step by step instructions that guide a novice user through a system

Visionary scenarios?

http://images.google.com/images?um=1&hl=en&q=before+after+picture

Before/After
www.makemeheal.com

Heros before/after
profile.myspace.com



Before and After Haircut
www.c71123.com

How do we find scenarios?

- Don’t expect the client to be verbal if the system does not exist (greenfield engineering)
- Don’t wait for information even if the system exists
- Engage in a ***dialectic*** approach (evolutionary, incremental engineering)
  - You help the client to formulate the requirements
  - The client helps you to understand the requirements
  - The requirements evolve while the scenarios are being developed

Heuristics for finding Scenarios

- Ask yourself or the client the following questions:
  - What are the primary tasks that the system needs to perform?
  - What data will the actor create, store, change, remove or add in the system?
  - What external changes does the system need to know about?
  - What changes or events will the actor of the system need to be informed about?

- However, don't rely on *questionnaires* alone.

- Insist on *task observation* if the system already exists (interface engineering or reengineering)
  - Ask to speak to the end user, not just to the software contractor
  - Expect resistance and try to overcome it

Example: Accident Management System

- What needs to be done to report a “Cat in a Tree” incident?
- What do you need to do if a person reports “Warehouse on Fire?”
- Who is involved in reporting an incident?
- What does the system do, if no police cars are available? If the police car has an accident on the way to the “cat in a tree” incident?
- What do you need to do if the “Cat in the Tree” turns into a “Grandma has fallen from the Ladder”?
- Can the system cope with a simultaneous incident report “Warehouse on Fire?”

How do you generate these questions?

Scenario Example: Warehouse on Fire

- Bob, driving down main street in his patrol car notices smoke coming out of a warehouse. His partner, Alice, ***reports the emergency*** from her car.

- Alice enters the address of the building, a brief description of its location (i.e., north west corner), and an emergency level. In addition to a fire unit, she requests several paramedic units on the scene given that area appear to be relatively busy. She confirms her input and waits for an acknowledgment.

- John, the Dispatcher, is alerted to the emergency by a beep of his workstation. He reviews the information submitted by Alice and acknowledges the report. He allocates a fire unit and two paramedic units to the Incident site and sends their estimated arrival time (ETA) to Alice.

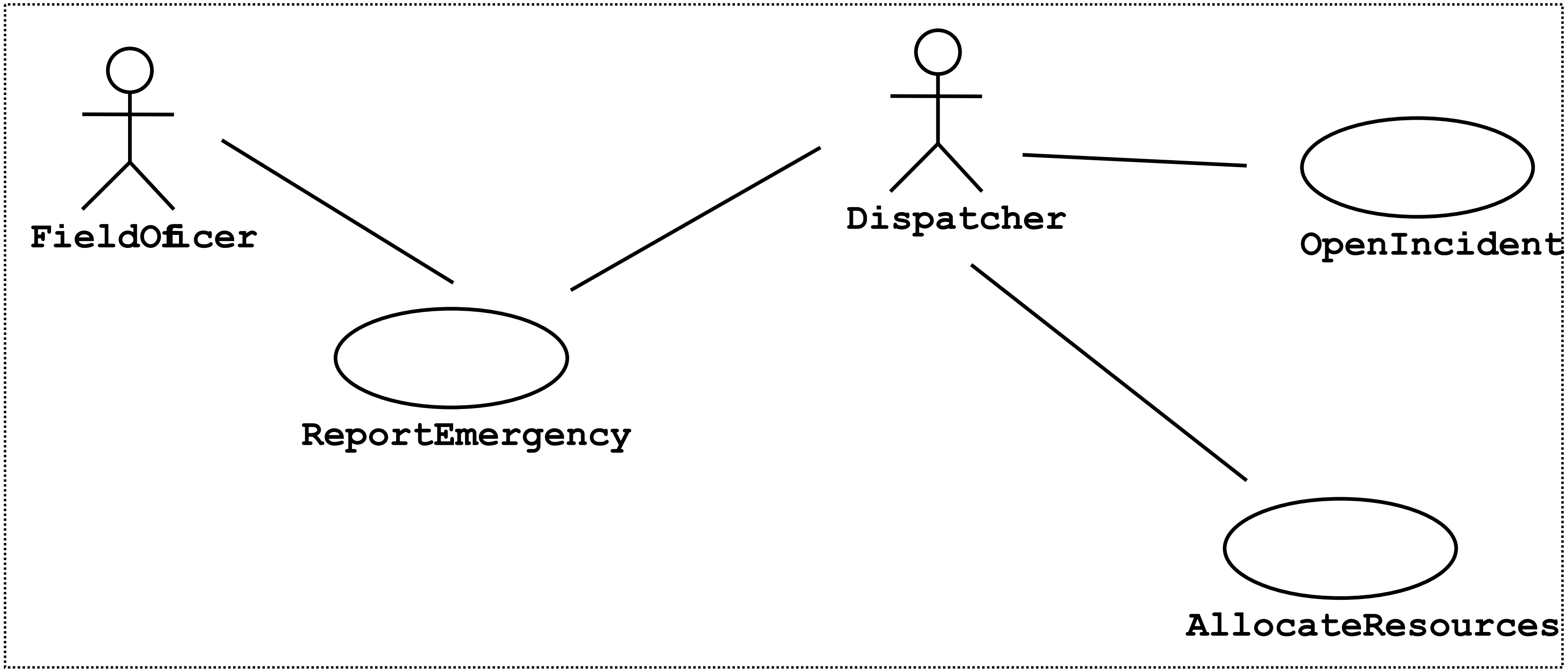
- Alice received the acknowledgment and the ETA.

Observations about Warehouse on Fire Scenario

- Concrete scenario
  - **Describes a single instance of reporting a fire incident.**
  - **Does not describe all possible situations in which a fire can be reported.**

- Participating actors
  - **Bob, Alice and John**

Example: Use Case Model for Incident Management



Use Case Example: ReportEmergency

Name of Use Case

Actors: Description of Actors involved in use case)

Entry condition: "This use case starts when…"

Flow of Events: Free form, informal natural language

Exit condition: "This use cases terminates when…"

Exceptions: Describe what happens if things go wrong

Special Requirements: NFR, Constraints

- Use case name: ReportEmergency
- Participating Actors:
  - Field Officer (Bob and Alice in the Scenario)
  - Dispatcher (John in the Scenario)
- Exceptions:
  - The FieldOfficer is notified immediately if the connection between her terminal and the central is lost.
  - The Dispatcher is notified immediately if the connection between any logged in FieldOfficer and the central is lost.
- Flow of Events: **on next slide.**
- Special Requirements:
  - The FieldOfficer's report is acknowledged within 30 seconds. The selected response arrives no later than 30 seconds after it is sent by the Dispatcher.

Use Case Example: ReportEmergency
Flow of Events

- The **FieldOfficer** activates the “Report Emergency” function of her terminal. FRIEND responds by presenting a form to the officer.

- The FieldOfficer fills the form, by selecting the emergency level, type, location, and brief description of the situation. The FieldOfficer also describes possible responses to the emergency situation. Once the form is completed, the FieldOfficer submits the form, at which point, the **Dispatcher** is notified.

- The Dispatcher reviews the submitted information and creates an Incident in the database by invoking the OpenIncident use case. The Dispatcher selects a response and acknowledges the emergency report.

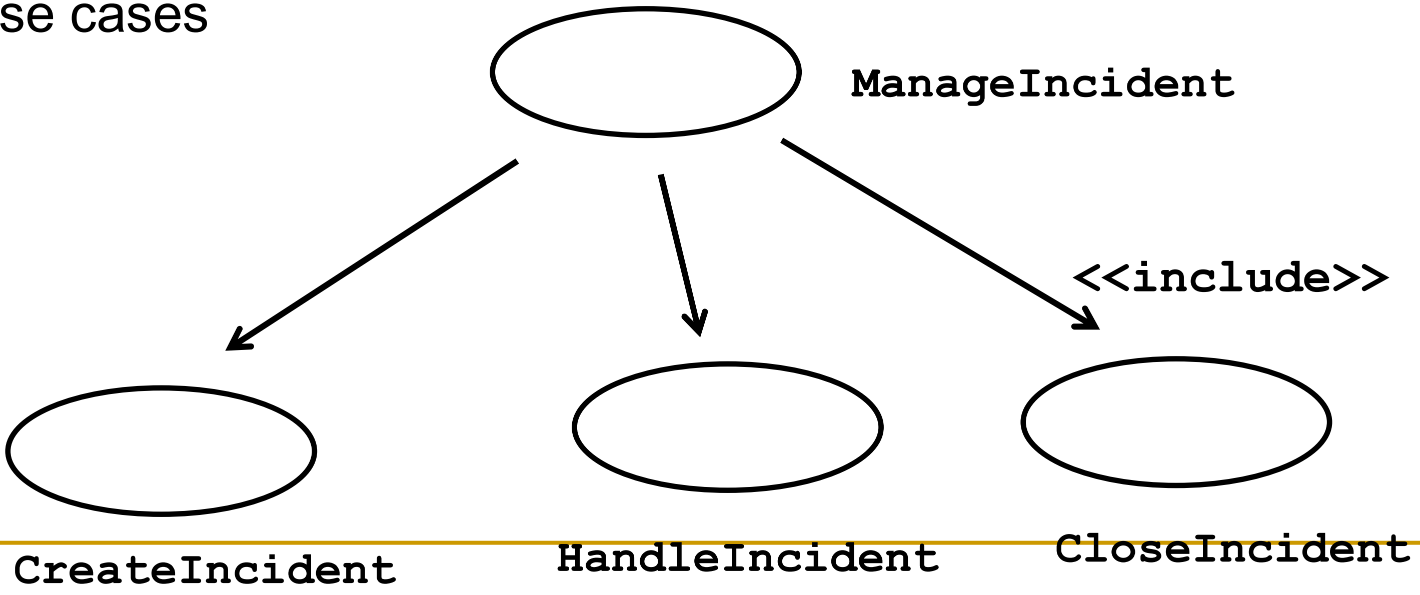
- The FieldOfficer receives the acknowledgment and the selected response.

Use Case Associations

- A use case model consists of use cases and use case associations
  - A use case association is a relationship between use cases
- Important types of use case associations: Include, Extends, Generalization
  - Include
    - A use case uses another use case ("functional decomposition")
  - Extends
    - A use case extends another use case
  - Generalization
    - An abstract use case has different specializations

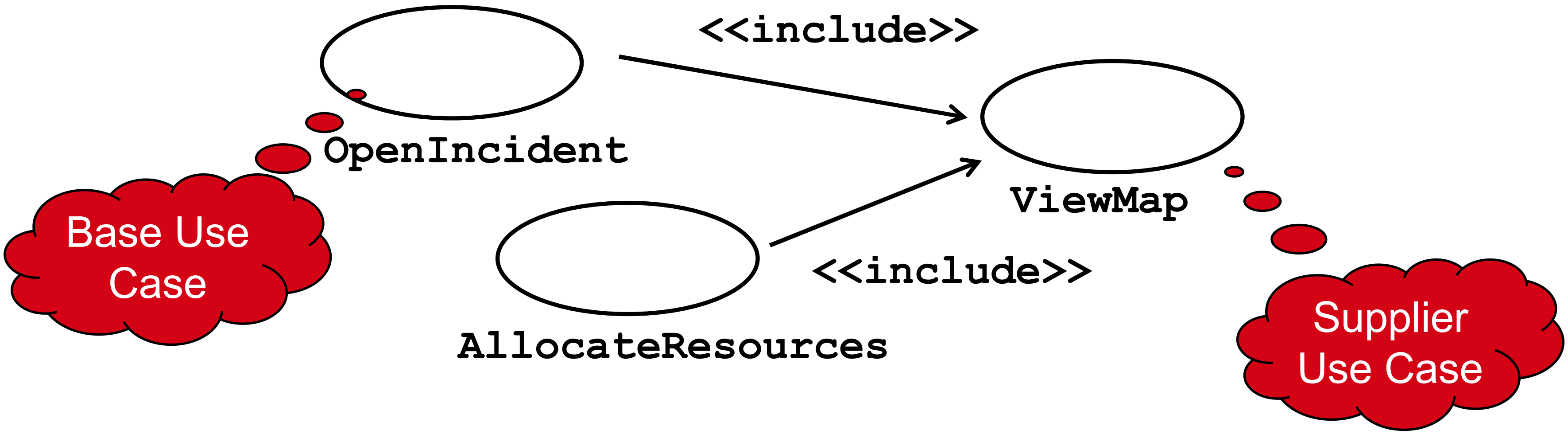
<<Include>>: Functional Decomposition

- Problem:
  - A function in the original problem statement is too complex to be solvable immediately
- Solution:
  - Describe the function as the aggregation of a set of simpler functions. The associated use case is decomposed into smaller use cases



<<Include>>: Reuse of Existing Functionality

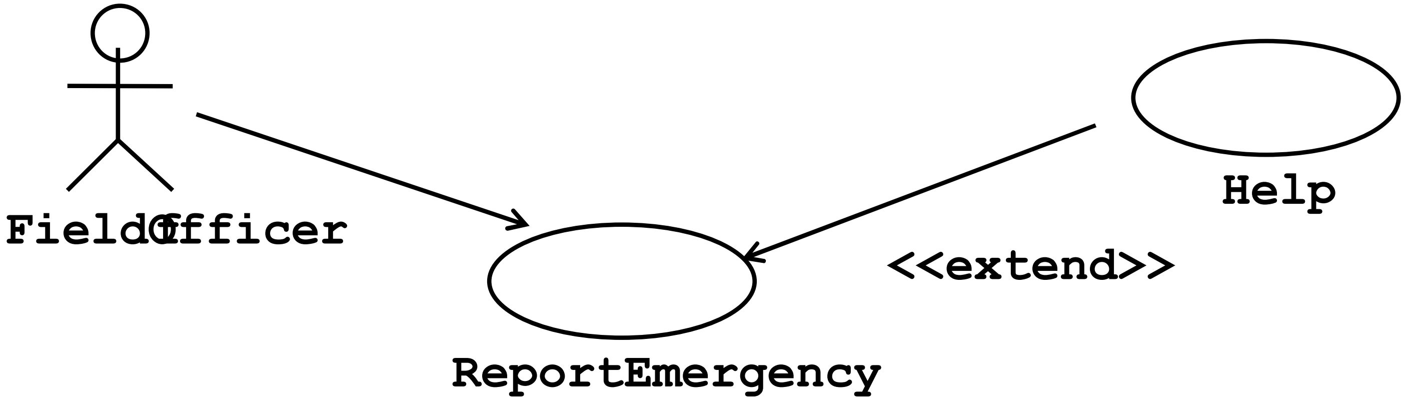
- Problem:
  - There are already existing functions. How can we *reuse* them?
- Solution:
  - The *include association* from a use case A to a use case B indicates that an instance of the use case A performs all the behavior described in the use case B ("A delegates to B")
- Example:
  - The use case "ViewMap" describes behavior that can be used by the use case "OpenIncident" ("ViewMap" is factored out)



Note: The base case cannot exist alone. It is always called with the supplier use case

<Extend>> Association for Use Cases

- Problem:
  - The functionality in the original problem statement needs to be extended.
- Solution:
  - An *extend association* from a use case A to a use case B indicates that use case B is an extension of use case A.
- Example:
  - The use case “ReportEmergency” is complete by itself , but can be extended by the use case “Help” for a specific scenario in which the user requires help



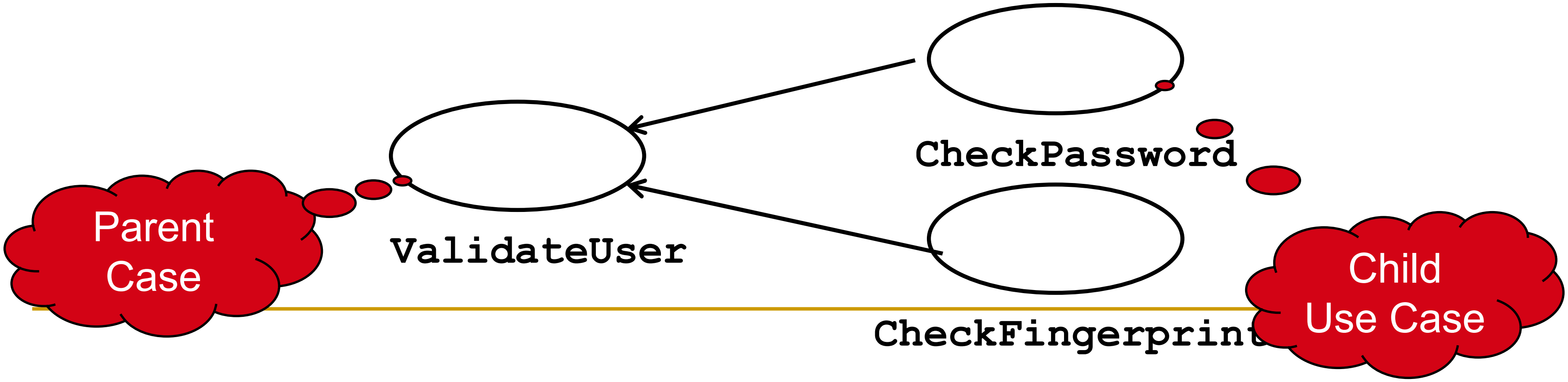
Note: The base use case can be executed without the use case extension in extend associations.

Generalization association in use cases

- Problem:
  - You have common behavior among use cases and want to factor this out.

- Solution:
  - The generalization association among use cases factors out common behavior. The child use cases inherit the behavior and meaning of the parent use case and add or override some behavior.

- Example:
  - Consider the use case “ValidateUser”, responsible for verifying the identity of the user. The customer might require two realizations: “CheckPassword” and “CheckFingerprint”



Scientific cycle of inquiry

[C. Potts, K. Takahashi, A. I. Antón, “Inquiry-Based Requirements Analysis,” *IEEE Software* 11(2): 21-32 (1994)]

- ***"Scenarios describe how users of the system interact with the system to achieve their particular tasks.“***

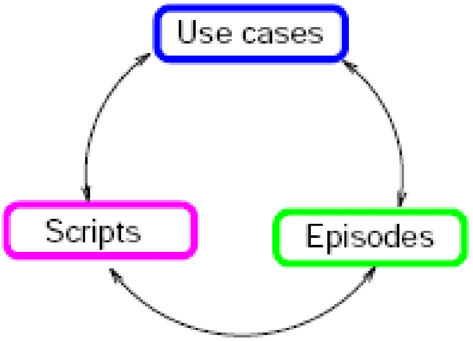
- ***"Scenarios usually refer to representative instances of user-system interactions."***

- **Types of Scenarios**
  - ***Use cases (paths):***
    - **short, informal descriptions of situations**
    - **possibly followed by explanatory phrases.**

  - ***Episodes:***
    - **in a tabular or diagrammatic form.**
    - **sequences of (detailed) user-system interaction**
    - **phases of activity;**
    - **an episode is a cluster of inter-related event occurrences.**

  - ***Scripts:***
    - ***(action table/diagram)***

- **CRC Cards**



Use Case/Paths

- **"A use case is a specific flow of events through the system (seen as a black box)“**

- ***Example***
  - A. The user case begins when the actor Guest enters the restaurant.
  - B. The actor Guest has the possibility of leaving her/his coat in the cloakroom, after which s/he is shown to a table and given a menu.
  - C. When the actor Guest has had sufficient time to make up her/his mind, s/he is asked to state her/his order. Alternatively, Guest can attract the waiter’s attention so that the order can be placed.
  - D. When the Guest has ordered, the kitchen is informed what food and beverages the order contains.
  - E. In the kitchen, certain basic ingredients, such as sauces, rice, and potatoes, have already been prepared. Cooking therefore involves collecting together these basic ingredients, adding spices and sorting out what needs to be done just before the dish is served.
  - F. When the dish is ready, it is served to the actor Guest. When it has been eaten, the actor is expected to attract the waiter’s attention in order to pay.
  - G. Once payment has been made, Guest can fetch her/his coat from the cloakroom and leave the restaurant. Then use case is then complete.

Use cases – in Usability Engineering & HCI

A task scenario is a sequence of system-user interactions and activities (a main basis for defect detection)

Example1

Label	Task Scenario #1
User Goal	To see if context switching is flexible in moving between searching and buying
Starting Point	The system is set-up, the user has entered the web site after login.
Intermediary Situation:	The user selects searching menu, but is unsure how to buy a chosen item.

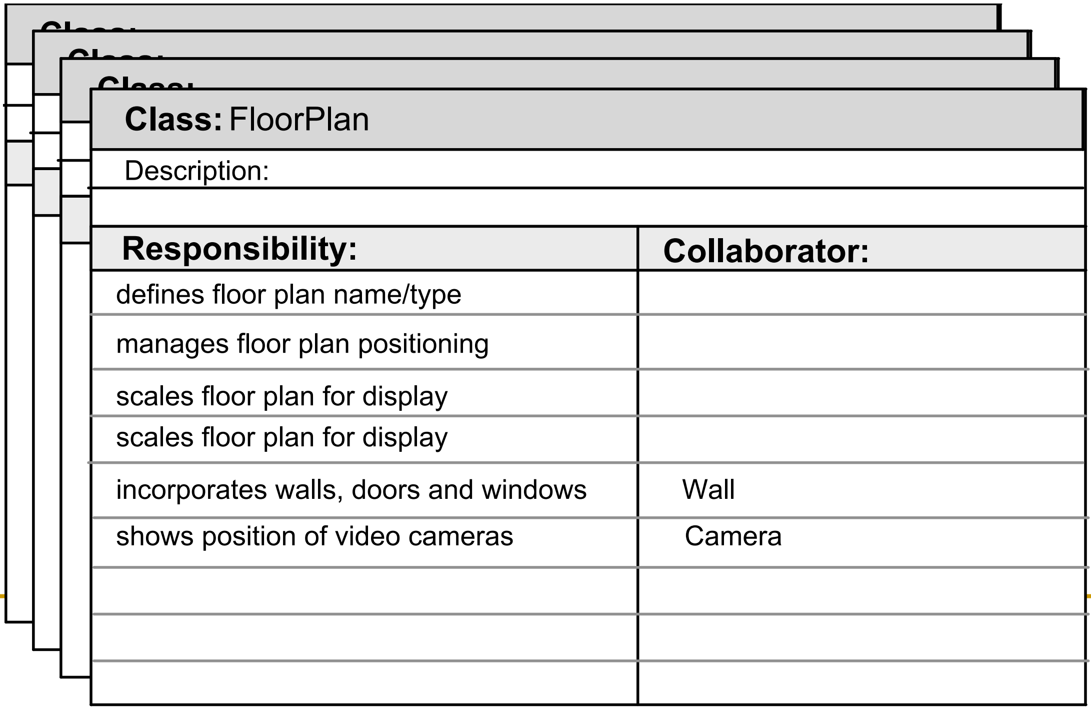
Example2

Label	Task Scenario #1
User Goal	To sign on to the web store
Starting Point	The system is set-up, the user has Entered the web site at the logon.
Intermediary Situation:	The user enters a user ID and no password.

CRC Modeling

- **Analysis classes have “responsibilities”**
  - *Responsibilities* are the attributes and operations encapsulated by the class

- **Analysis classes collaborate with one another**
  - *Collaborators* are those classes that are required to provide a class with the information needed to complete a responsibility.
  - In general, a collaboration implies either a request for information or a request for some action.



Class:

Class:

Class:

Class: FloorPlan

Description:

Responsibility:	**Collaborator:**
defines floor plan name/type	
manages floor plan positioning	
scales floor plan for display	
scales floor plan for display	
incorporates walls, doors and windows	Wall
shows position of video cameras	Camera

Example: Meeting schedule system

- *Conflicts*
- *reminders to late {participants, resource manager}*
- *Confirmation*
- *changes in {preference, exclusion} sets*
- *Dropout*
- *cancel meeting*
- *reschedule meeting*
- *changes of active participants and/or equipments*
- *substitute active and/or important participants*
- *multiple booking*
- *meeting bumped by more important meeting*

Example: Meeting Schedule

Episodes

- **Episode1: Initiation**
  - Initiator determines important, active & other participants
  - Initiator asks important participants for location prefs.
  - Initiator asks active participants for location preferences
  - Initiator asks active participants for equipment reqs
  - Initiator asks for exclusion set from potential participants
  - Initiator asks for preference set from potential participants
  - Initiator prescribes date range
- **Episode2: Responding**
  - Participants respond to requests for pref. & excl. sets
  - Active participants respond for equipment reqs.
  - Important participants responds for preferred loc.
- **Episode3: Scheduling**
  - Scheduler chooses meeting time
  - Scheduler chooses location
- **Episode4: Reserving & Notification**
  - Scheduler reserves room & equipment
  - Scheduler notifies participants & initiator of meeting

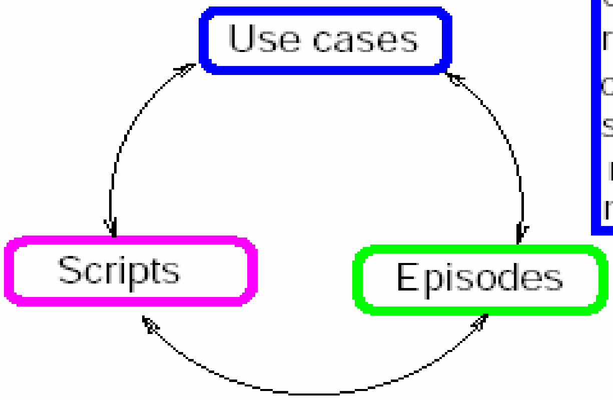
A high-level script

- Initiator prescribes date range
- Initiator determines participants
- Initiator asks for needed information
- Participants respond
- Scheduler sets meeting & informs everybody

Use cases Episodes Scripts

Sample use cases

conflicts
reminders to late {parcipants, resource manager}
confirmation
changes in {preference, exclusion} sets
dropout
cancel meeting
reschedule meeting
changes of active participants and/or equipments
substitute active and/or important participants
multiple booking
meeting bumped by more imprtant meeting



Episode1: Initiation

Initiator determines important, active & other participants
Initiator prescribes date range
Initiator asks for preference set from potential participants
Initiator asks for exclusion set from potential participants
Initiator asks active participants for equipment reqs
Initiator asks active participants for location preferences
Initiator asks important participants for location prefs.

Episode2: Responding

Participants respond to requests for pref. & excl. sets
Active participants respond for equipment reqs.
Important participants responds for preferred loc.

Episode3: Scheduling

Scheduler chooses meeting time
Scheduler chooses location

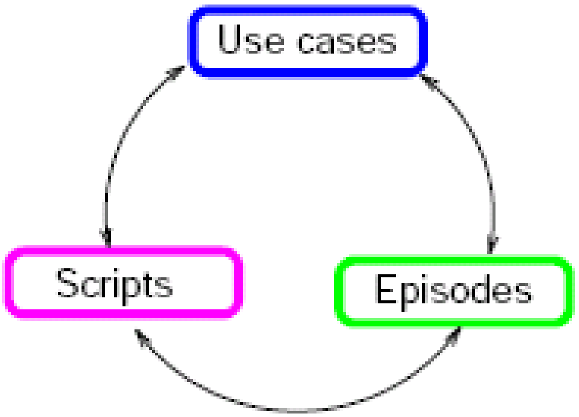
Episode4: Reserving & Notification

Scheduler reserves room & equipment
Scheduler notifies participants & initiator of meeting

A high-level script

Initiator determines participants
Initiator prescribes date range
Initiator asks for needed information
Participants respond
Scheduler sets meeting &
inform everybody

Use cases, Episodes, Scripts



A high-level script b

Initiator determines participants
Initiator prescribes date range
Scheduler asks for needed information
Participants respond
Scheduler sets meeting &
inform everybody

Episode1: Initiation

Initiator determines important, active & other participants
Initiator prescribes date range

Episode1b: Initiation by Scheduler

Scheduler asks for preference set from potential participants
exclusion set from potential participants
active participants for equipment reqs
active participants for location preferences
important participants for location prefs.

Episode2: Responding

Participants respond to requests for pref. & excl. sets
Active participants respond for equipment reqs.
Important participants responds for preferred loc.

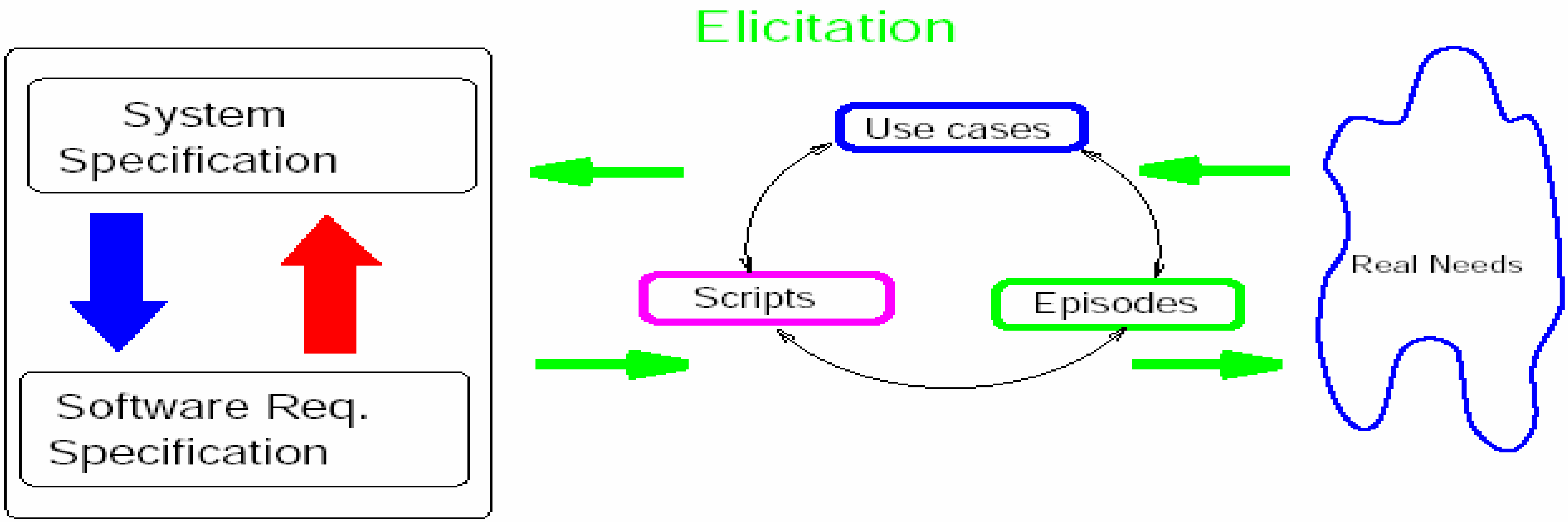
Episode3: Scheduling

Scheduler chooses meeting time
Scheduler chooses location

Episode4: Reserving & Notification

Scheduler reserves room & equipment
Scheduler notifies participants & initiator of meeting

Scenario Analysis: How?



- *Go through an iterative process of refinement:*
  - Challenge requirements (specifications)
  - Add use cases, episodal structures & conjoin them to construct scripts
  - Challenge scenarios

Scenario Instantiation - Script for: Alice Goes To Wonderland

Initiator determines important, active & other participants

Ian Martha Alice Olga

Initiator prescribes date range

Mon-Fri next week (it's Monday p.m. now)

Scheduler asks for preference set from potential participants

Martha Alice Olga

Scheduler asks for exclusion set from potential participants

Scheduler asks active participants for equipment reqs

Alice

Scheduler asks active participants for location preferences

Scheduler asks important participants for location prefs.

Martha

Participants respond to requests for pref. & excl. sets

Martha	Tue-Fri afternoon	Mon morning
Alice		Mon-Fri next week
Olga	Mon-Thu	Fri afternoon

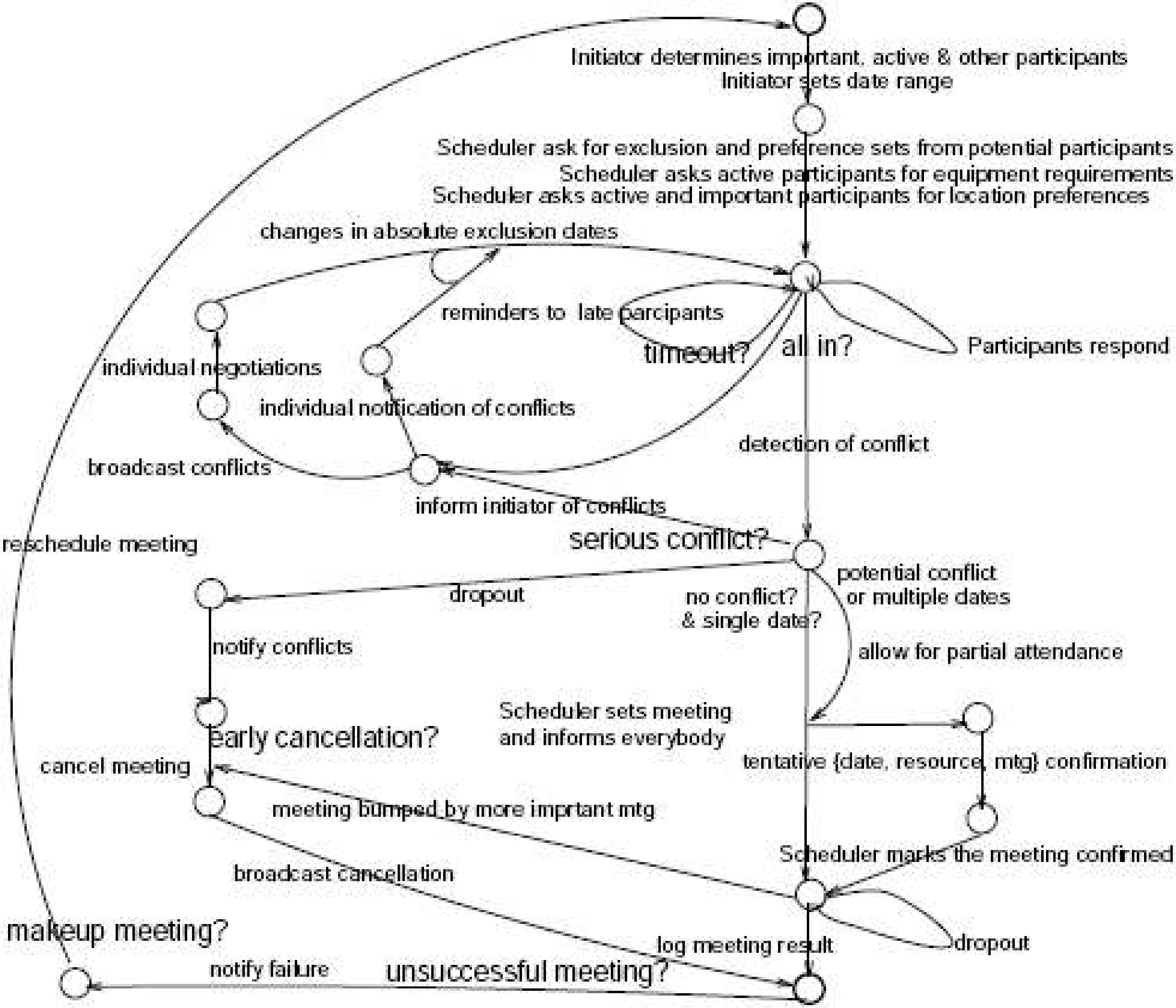
Active participants respond for equipment reqs.
Important participants responds for preferred loc.

Scheduler chooses meeting time
Scheduler chooses location

Scheduler reserves room & equipment
Scheduler notifies participants & initiator of meeting

Conflict -> new episodes, use cases, scripts

Detailed Script (Action Diagram)


Initiator determines important, active & other participants
Initiator sets date range
Scheduler ask for exclusion and preference sets from potential participants
Scheduler asks active participants for equipment requirements
Scheduler asks active and important participants for location preferences
changes in absolute exclusion dates
reminders to late parcipants
timeout?
all in?
Participants respond
individual negotiations
individual notification of conflicts
broadcast conflicts
detection of conflict
inform initiator of conflicts
serious conflict?
reschedule meeting
potential conflict
or multiple dates
dropout
no conflict?
& single date?
notify conflicts
allow for partial attendance
Scheduler sets meeting
and informs everybody
early cancellation?
cancel meeting
tentative {date, resource, mtg} confirmation
meeting bumped by more imprtant mtg
Scheduler marks the meeting confirmed
broadcast cancellation
makeup meeting?
log meeting result
dropout
notify failure
unsuccessful meeting?

From As-Is to To-Be

“Scenarios help define the system”

“But, potentially an infinite number of scenarios“

“Hence, any scenario set is necessarily incomplete“

“But the quality of scenarios is correlated to the quality of the system”

So, how should we explore scenarios
so that a good system may be defined?

Success of Scenarios

- ***Largely depends on how well questions are posed***
  - but, generation of questions can be hard
    - categorize question types
    - select and pose questions systematically, while also using bottom-up approach

- ***Answers to questions:***
  - might be given by the analyst and users through analysis of scenarios
    - refinements of scenarios
    - refinements of existing requirements
    - discovery of missing requirements

**Criteria: How good, and how cost-effective, is our understanding of
the problem, goals, domain knowledge, requirements and specification**

(How do we understand the problem, goals, domain knowledge, requirements and specification?)

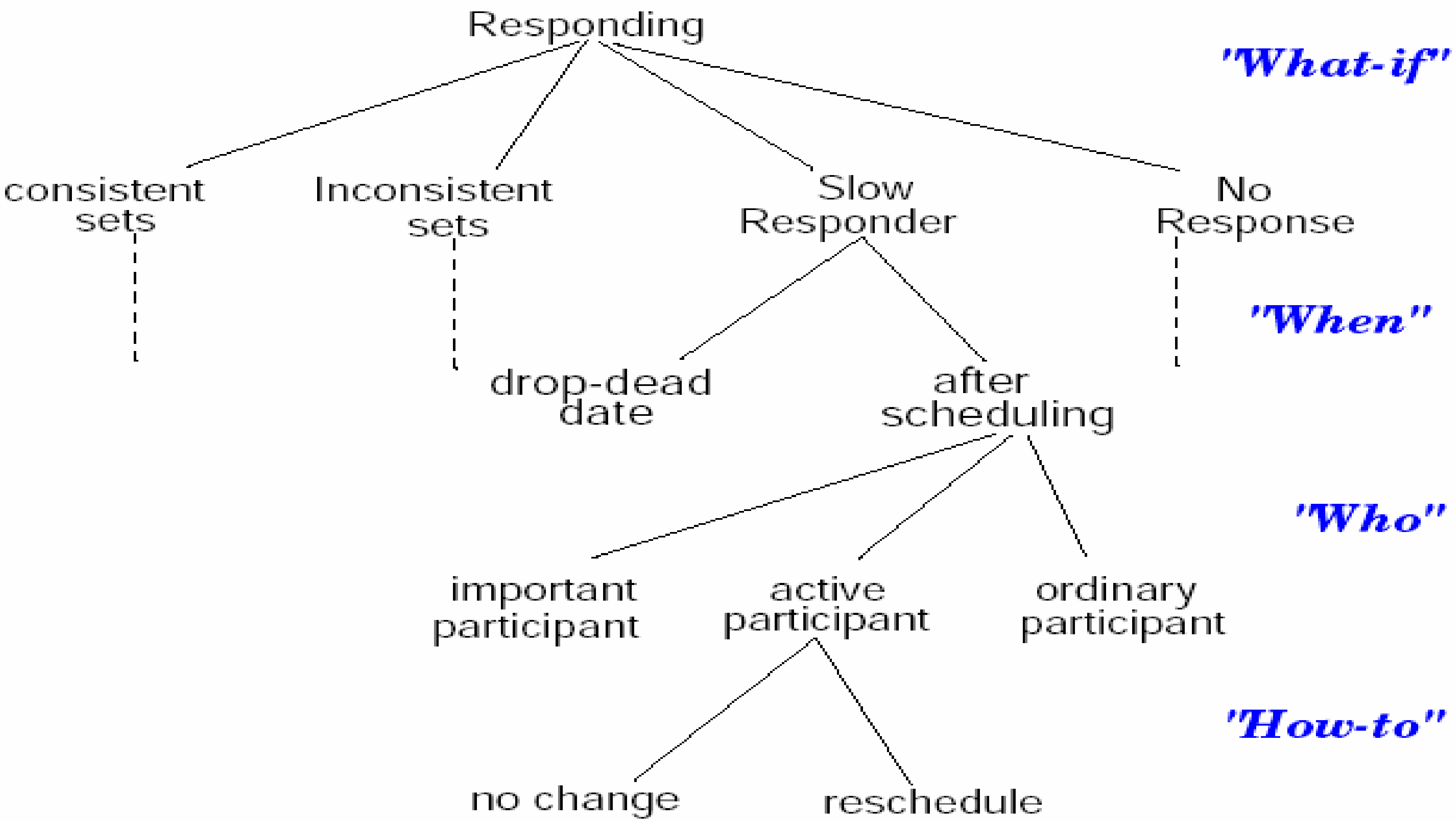
Question Types

- ***"What-if":***
  - pursue hypothetical "what could go wrong?" lines of reasoning
  - "What could go wrong with participants’ response to the date set?"
    - (i) Participants submit consistent preferred date set
    - (ii) Participants submit inconsistent preferred & exclusion *(Inconsistent sets)*
    - (iii) Participants submit preferred date set late *(Slow Responder)*
    - (iv) Participants do not submit preferences *(No Response)*
- ***"Who":***
  - "Who initiates a meeting?"
    - (i) An initiator (Person)
    - (ii) An initiator (Person) & The Meeting Scheduler
    - (i) The Meeting Scheduler

Questions Continued

- *"What-kinds-of":*
  - "What kinds of meeting should be supported?"
    - (i) One-shot meeting
    - (ii) Periodic meeting
- *"When:*
  - "If a potential meeting attendee does not respond, at what time should the scheduler go ahead & schedule the meeting?"
    - ***drop-dead date***
- *"How-to":*
  - "How can participants resolve conflicts?“
    - (i) The Meeting Scheduler
    - (ii) An initiator (Person) & The Meeting Scheduler

Tree based analysis



Success of Scenarios:

"Potentially endless chains of questions & answers"
"Potentially an infinite number of scenarios"
"Any scenario set is necessarily incomplete“

- ***Prioritize scenarios (/requirements)***
  - critical scenarios (benefits, costs, risks)
  - high frequency scenarios

- ***Pruning***
  - divide the scenario space into 3 mutually exclusive sets:
    - *discarded set: of scenarios which won’t be considered further*
    - *selected set:*
    - *undecided set:*

- ***Explore scenarios based on priorities***
  - use breadth-first or depth-first accordingly

London Ambulance Case Study

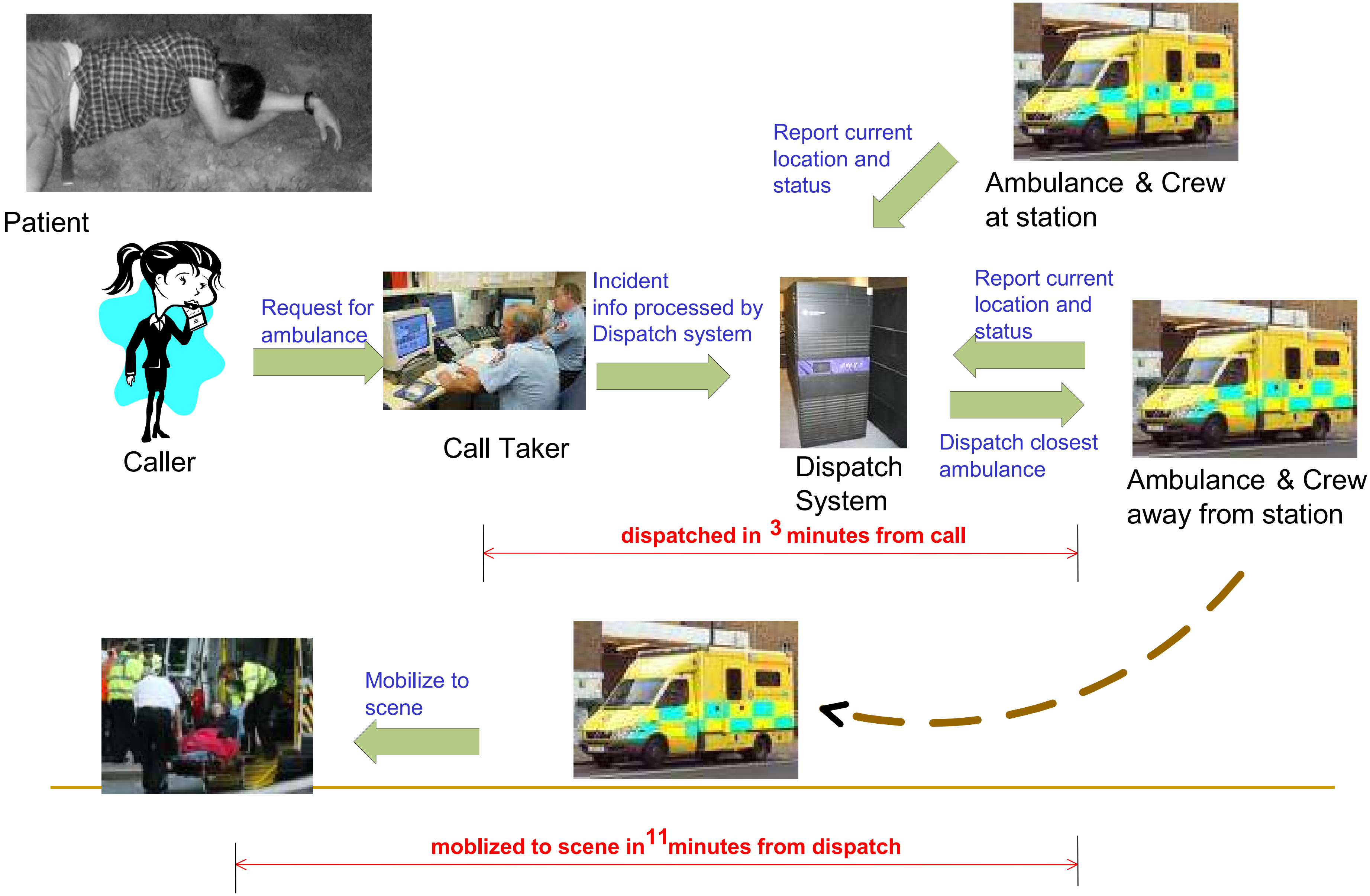
Adopted from Supakkul's working memo: problems behind problem frames

- A Computer-Aided-Dispatch (CAD) system deployed in October, 1992
- Business goal: to meet the new regulation:
  - Ambulance arrives in 14 minutes
    - Dispatched in 3 mins. from the call
    - Arrived at scene in 11 mins.

- System function:
  - Automate the tracking and dispatching of ambulances

London Ambulance 1992 dispatch system:

Successful scenario



Patient

Caller

Request for ambulance

Call Taker

Incident info processed by Dispatch system

Dispatch System

Report current location and status

Ambulance & Crew at station

Report current location and status

Dispatch closest ambulance

Ambulance & Crew away from station

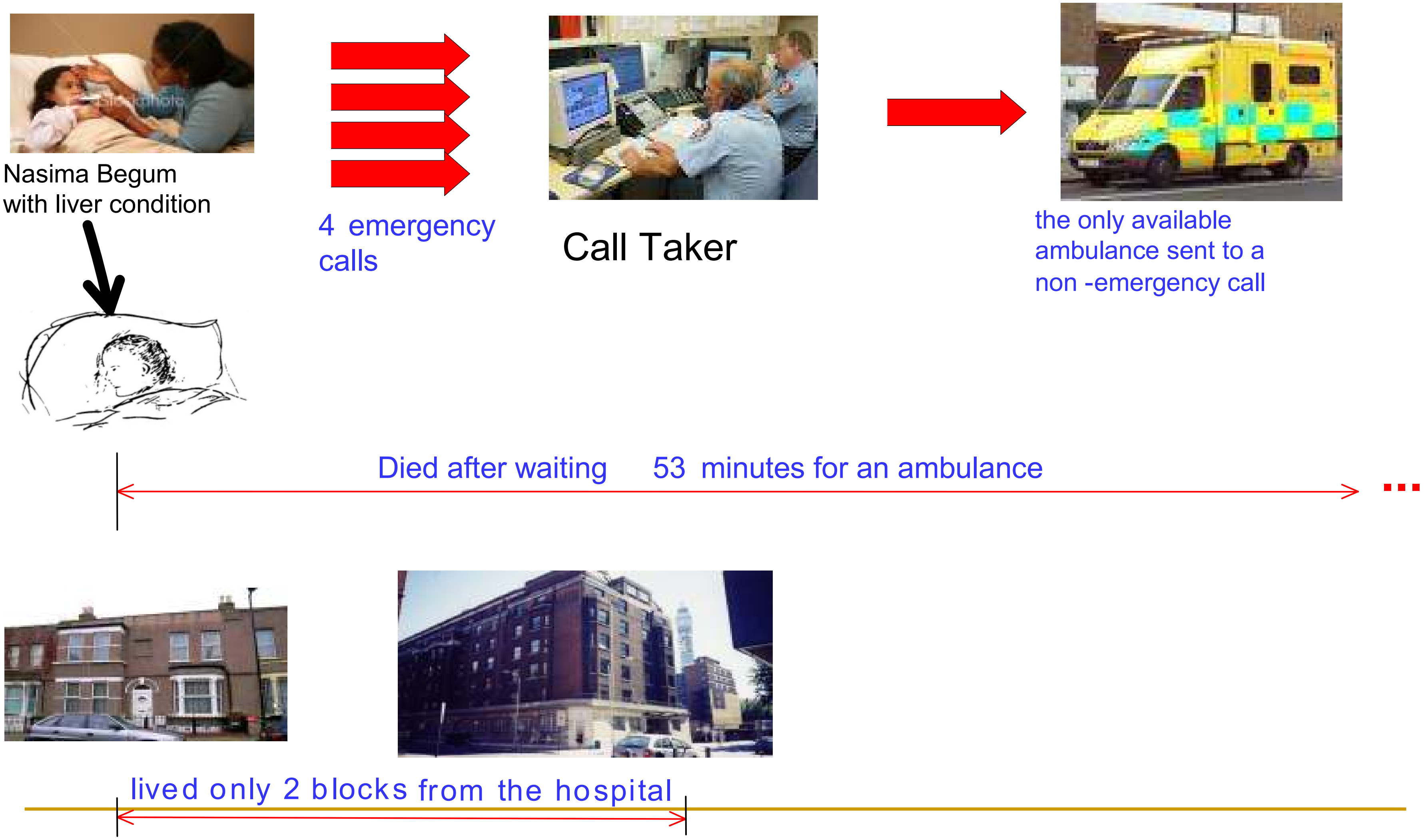
dispatched in 3 minutes from call

Mobilize to scene

moblized to scene in 11 minutes from dispatch

London Ambulance 1992 dispatch system:

Nasima Begum mishap scenario



Nasima Begum with liver condition

4 emergency calls

Call Taker

the only available ambulance sent to a non -emergency call

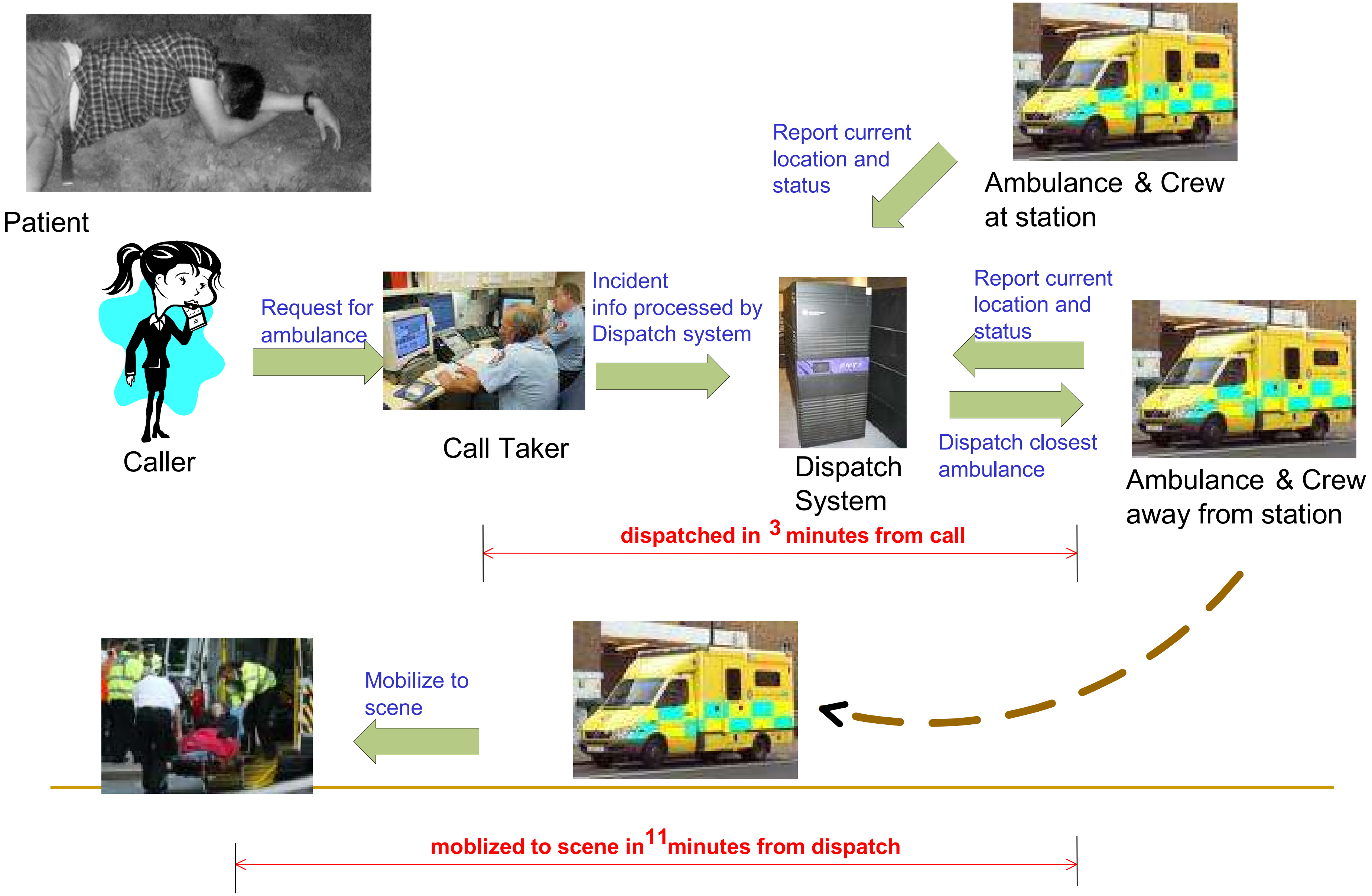
Died after waiting 53 minutes for an ambulance ...

lived only 2 blocks from the hospital

Note: some source (Guy Fitzgeral's "The Turnaround of the London Ambulance…" indicated this incident occurred after LAS went back to use the manual dispatch in June 1994 after the mishap in 1992 while some source cited this incident in the 1992 mishap (is it D. Dalcher's "Disaster…"?).

London Ambulance 1992 dispatch system:

Successful scenario



Patient

Caller

Request for ambulance

Call Taker

Incident info processed by Dispatch system

Dispatch System

Report current location and status

Ambulance & Crew at station

Report current location and status

Dispatch closest ambulance

Ambulance & Crew away from station

dispatched in 3 minutes from call

Mobilize to scene

moblized to scene in 11 minutes from dispatch

London Ambulance 1992 dispatch system:
Nasima Begum mishap scenario

Nasima Begum
with liver condition

4 emergency
calls

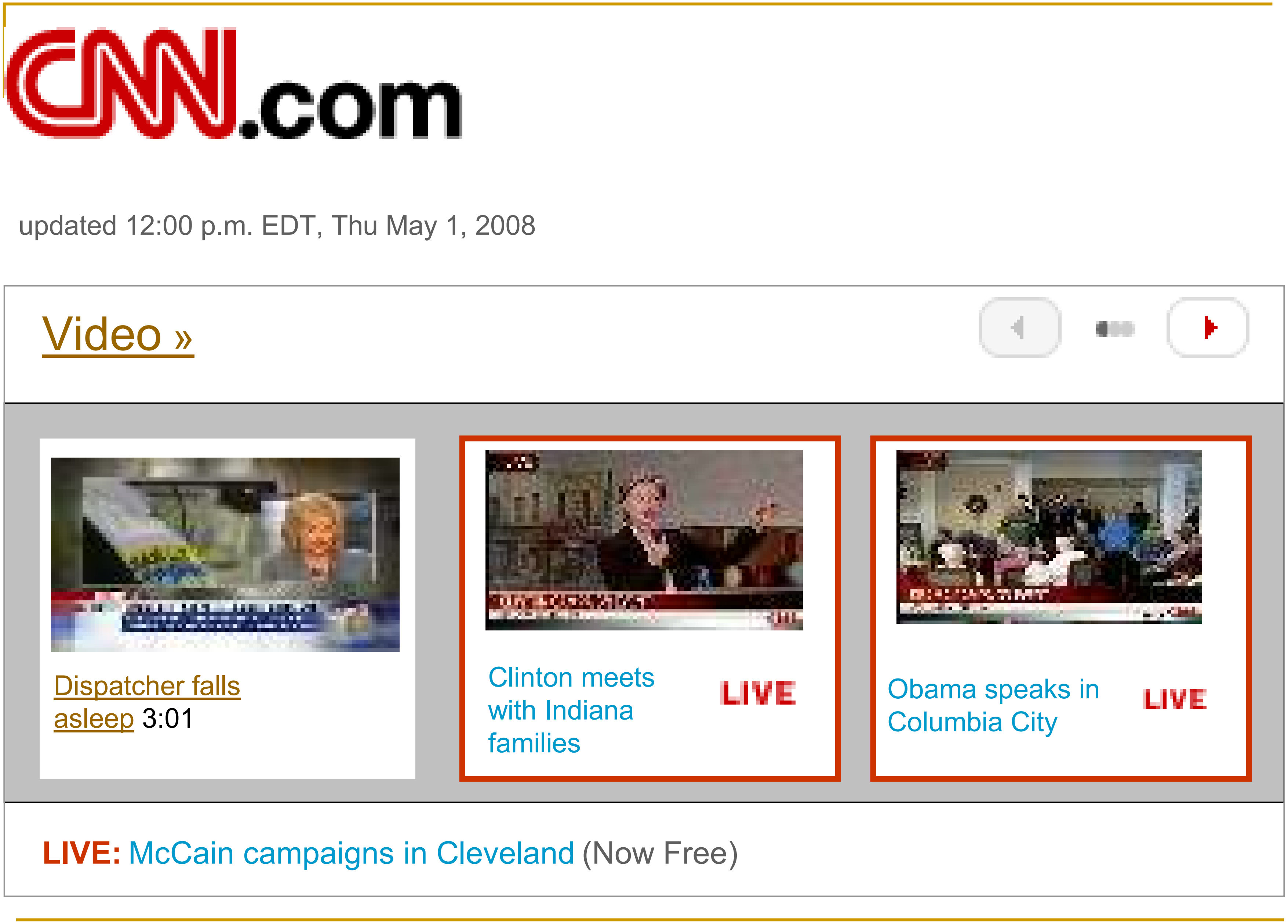
Call Taker

the only available
ambulance sent to a
non -emergency call

Died after waiting 53 minutes for an ambulance ...

lived only 2 blocks from the hospital

Note: some source (Guy Fitzgeral's "The Turnaround of the London Ambulance…" indicated this incident occurred after LAS went back to use the manual dispatch in June 1994 after the mishap in 1992 while some source cited this incident in the 1992 mishap (is it D. Dalcher's "Disaster…"?).



updated 12:00 p.m. EDT, Thu May 1, 2008

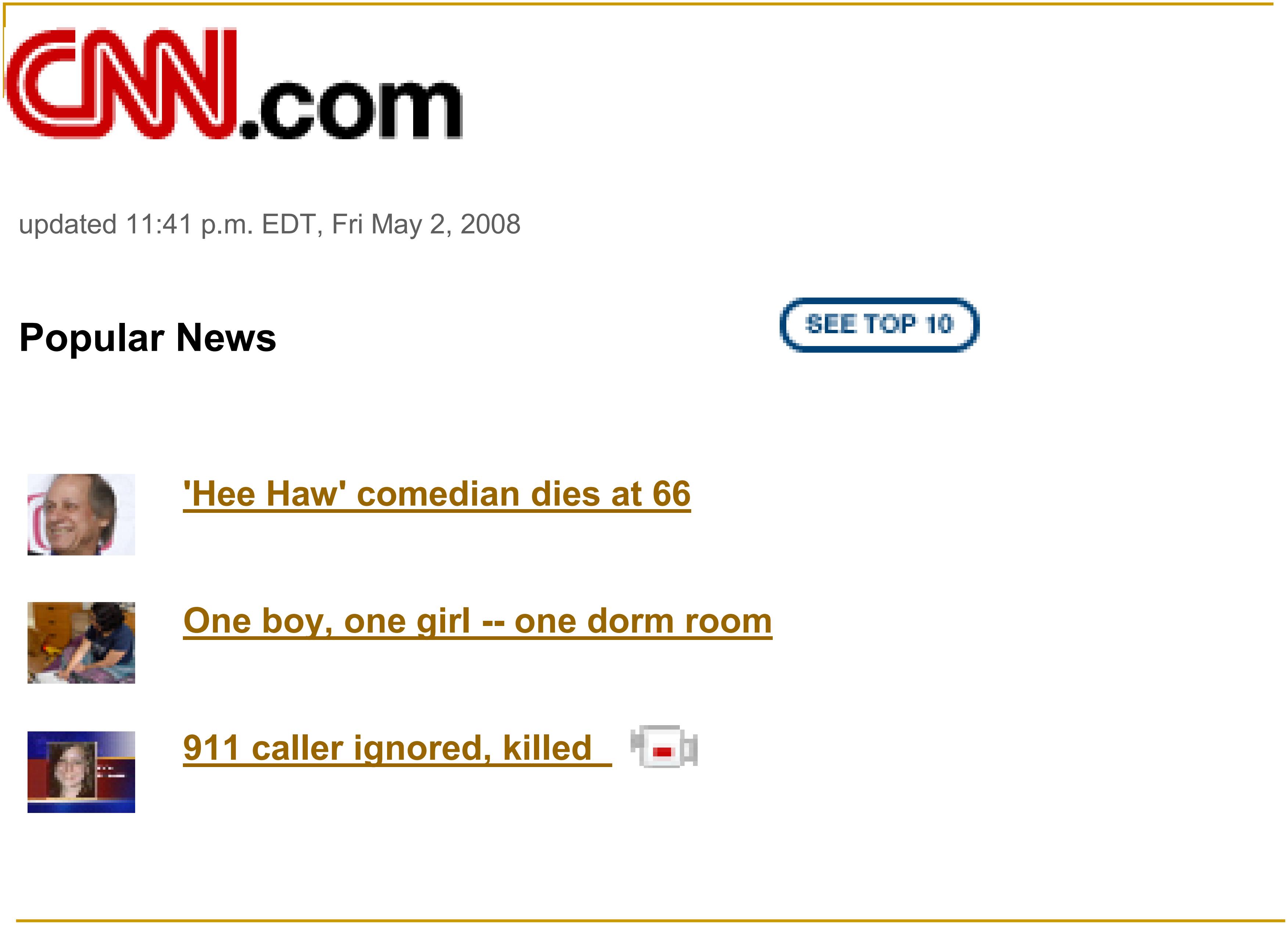
Video »

Dispatcher falls asleep 3:01

Clinton meets with Indiana families LIVE

Obama speaks in Columbia City LIVE

LIVE: McCain campaigns in Cleveland (Now Free)



CNN.com

updated 11:41 p.m. EDT, Fri May 2, 2008

Popular News

'Hee Haw' comedian dies at 66

One boy, one girl -- one dorm room

911 caller ignored, killed

What's the problem?

London Ambulance Case Study

- A Computer-Aided-Dispatch (CAD) system deployed in October, 1992

- Business goal: to meet the new regulation:
    - Ambulance arrives in 14 minutes
        - Dispatched in 3 mins. from the call
        - Arrived at scene in 11 mins.

- System function:
    - Automate the tracking and dispatching of ambulances

➢**Many died from not getting care in time:**

- An 11-year old girl died of a kidney condition after waiting for 53 mins

- A man died of heart attack after waiting for 2 hours

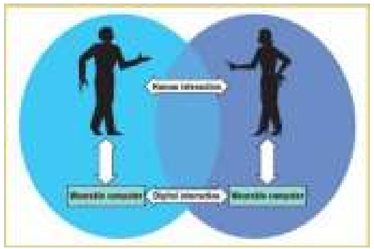
➢**Multiple ambulances sent to the same incident**

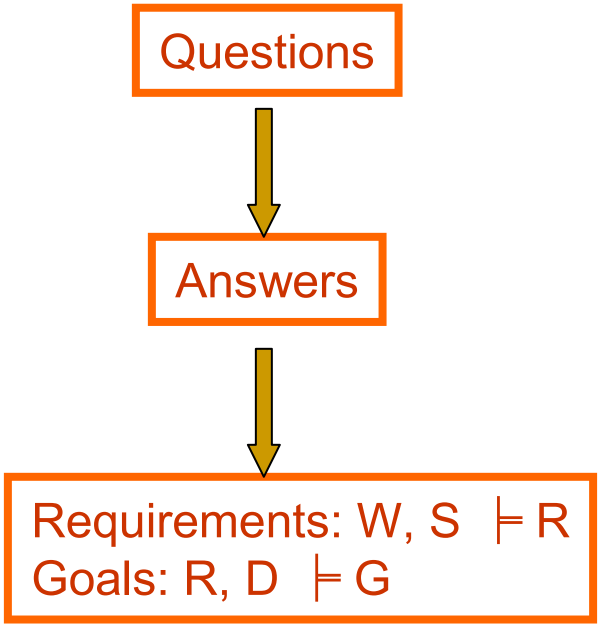
➢**Lost track of the ambulance status such that operator had to call the caller back to check if an ambulance had arrived**

A workshop in software engineering concluded: ***NFRs (non-functional requirements) were not considered early in the development process,*** among other organizational and software engineering mistakes

Successful Scenario Analysis

UbiCom for Humans thru Collaboration





Questions

Answers

Requirements: W, S ⊨ R
Goals: R, D ⊨ G

Who dispatches ambulance?

A dispatcher dispatches ambulance

How?

by communicating with ambulance driver

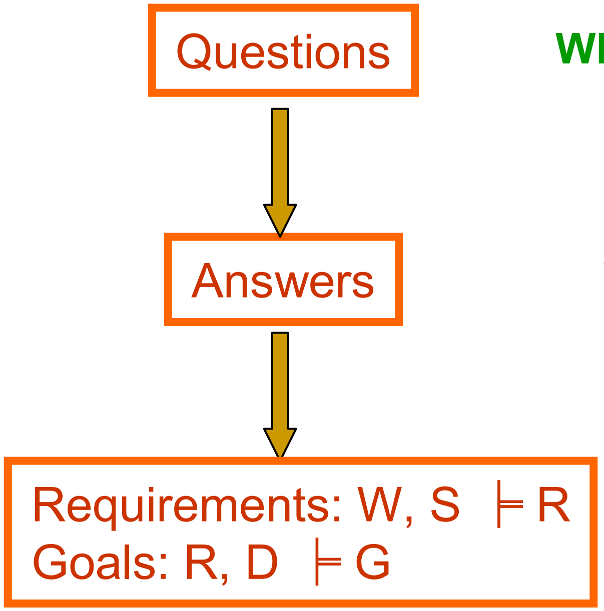
How can questions be phrased so that answers become relevant and complete wrt. requirements?

(How do we understand the problem, goals, domain knowledge, requirements and specification?)

Successful Scenario Analysis

UbiCom for Humans thru Collaboration

Model-driven approach



Requirements: W, S $\models$ R

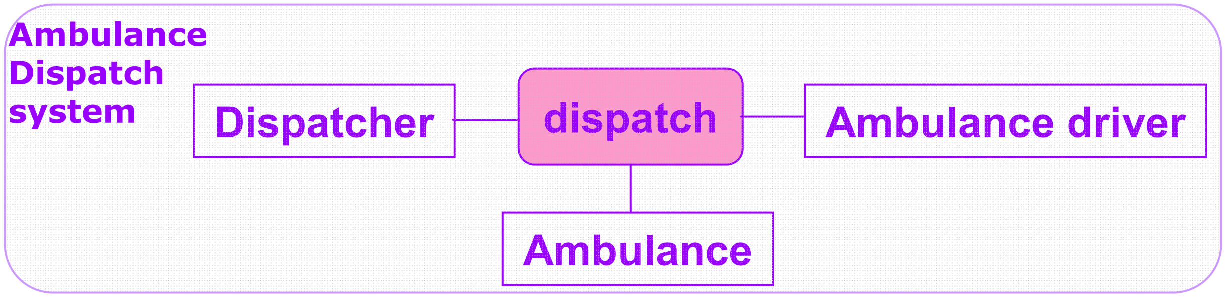
Goals: R, D $\models$ G

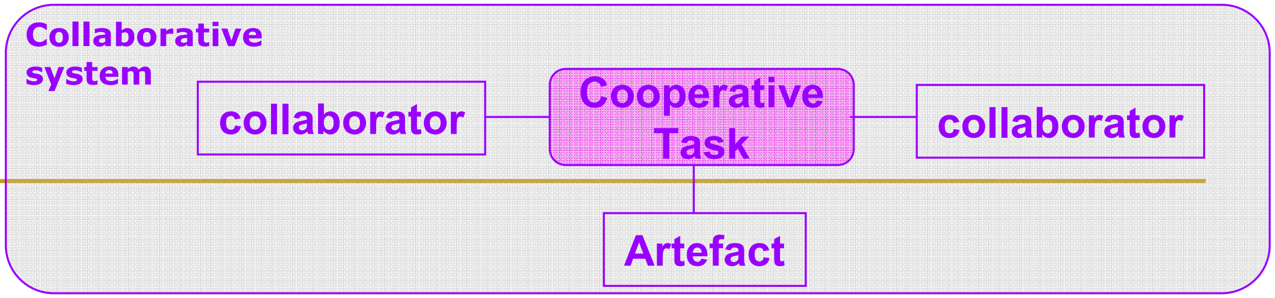
Who dispatches ambulance?

A dispatcher dispatches ambulance

How?

by communicating with ambulance driver





Question Types

- **What-if:** pursue hypothetical "what could go wrong?" lines of reasoning
  - What if no ambulance is available?
    - (i) Can a nearby ambulance accommodate another patient?
    - (ii) Can an ambulance assigned to a less critical incident be re-allocated?

- **Who:**
  - Who dispatches ambulance? (i) A dispatcher; (ii) a call-taker; (iii) a software agent
  - Who should be notified? (i) 911; (ii) emergency contact; (iii) fire department; (iv) hospital; (v) MD

- **What-kind-of:**
  - What kind of ambulance should be dispatched?
    - (i) only with basic emergency kit
    - (ii) mobile hospital with qualified crew

- **When:**
  - When should an emergency contact be notified? (i) immediately; (ii) at 911; (iii) at ambulance …

- **How-to:**
  - How to obtain patient information?“ (i) observation; (ii) RFID tag
  - How to determine patient location? (i) caller knowledge; (ii) GPS; (iii) distributed location indicator
  - How to transfer information? (i) verbal; (ii) push-button mobile device; (iii) partial, semi-automatic

- **Why**

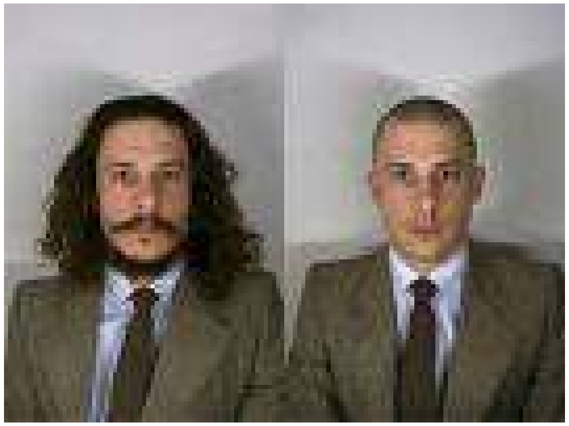
- **…**

Visionary scenarios: *how?*

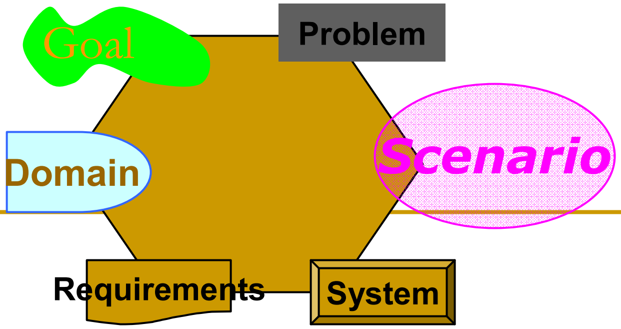
http://images.google.com/images?um=1&hl=en&q=before+after+picture

Before/After
www.makemeheal.com

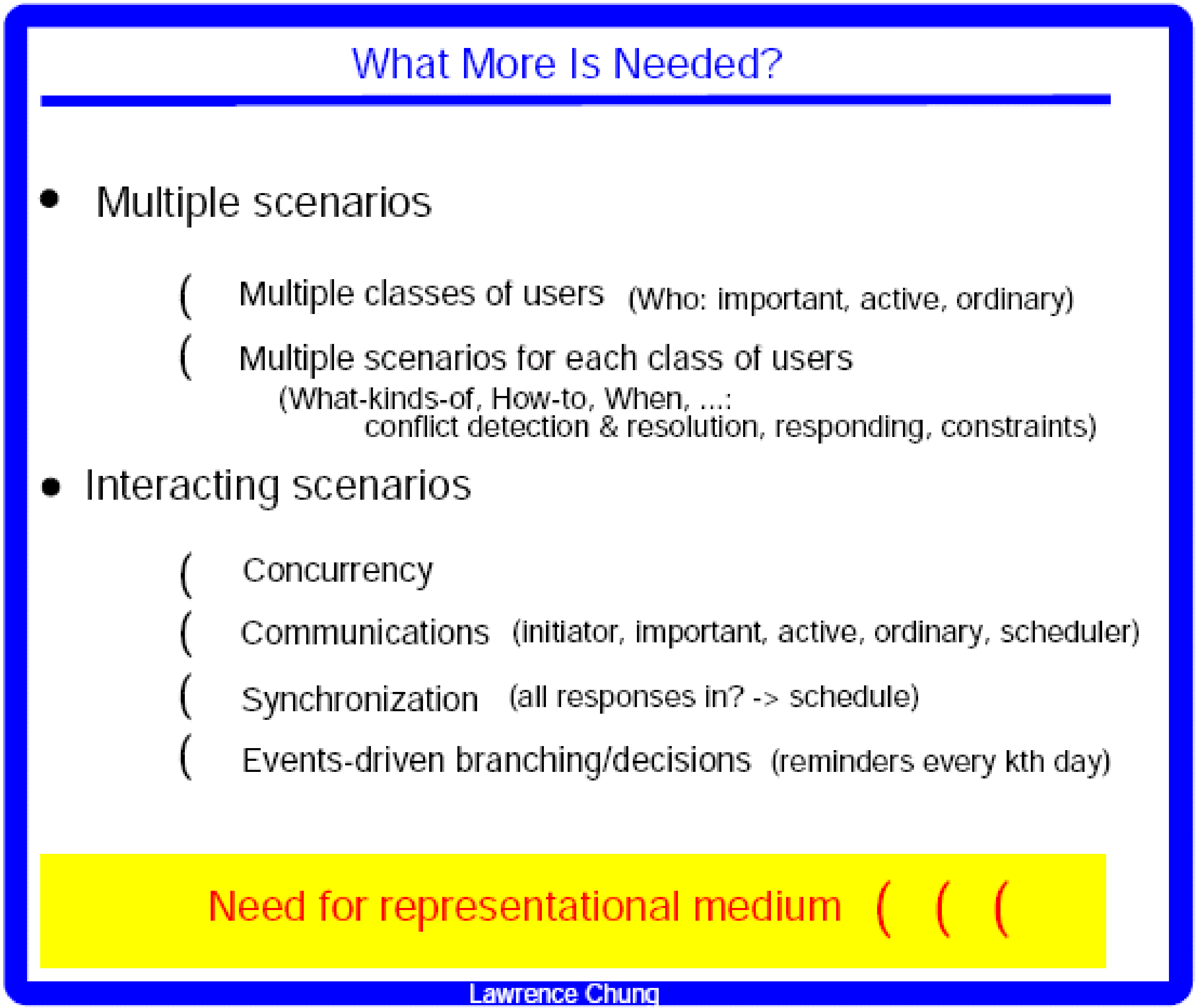
Heros before/after
profile.myspace.com



Before and After Haircut
www.c71123.com



Appendix



What More Is Needed?

- Multiple scenarios

  (Multiple classes of users (Who: important, active, ordinary)

  (Multiple scenarios for each class of users
  (What-kinds-of, How-to, When, ...:
  conflict detection & resolution, responding, constraints)

- Interacting scenarios

  (Concurrency

  (Communications (initiator, important, active, ordinary, scheduler)

  (Synchronization (all responses in? -> schedule)

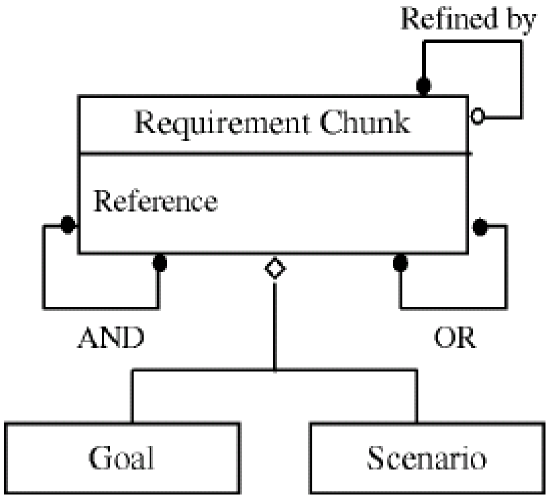
  (Events-driven branching/decisions (reminders every kth day)

Need for representational medium (((

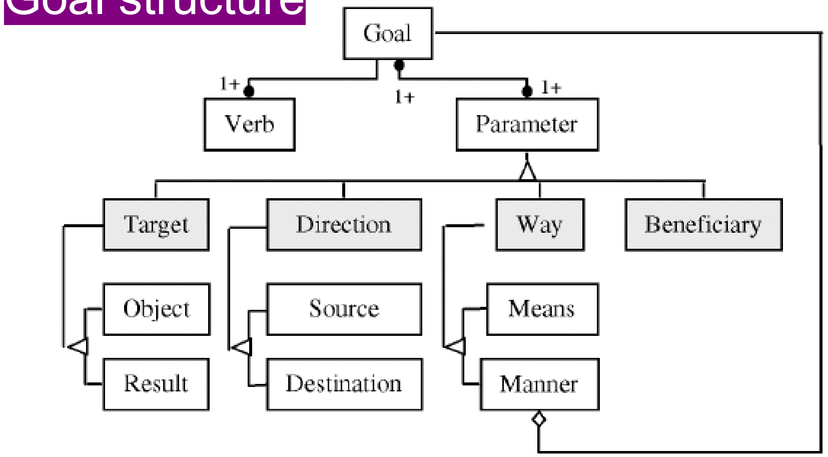
Goals & Scenarios

C. Rolland, C. Souveyet, and C. Ben Achour, “Guiding Goal Modeling Using Scenarios,”
IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 24, NO. 12, Dec. 1998. pp. 1055-1071.

Requirements chunk model



Goal structure



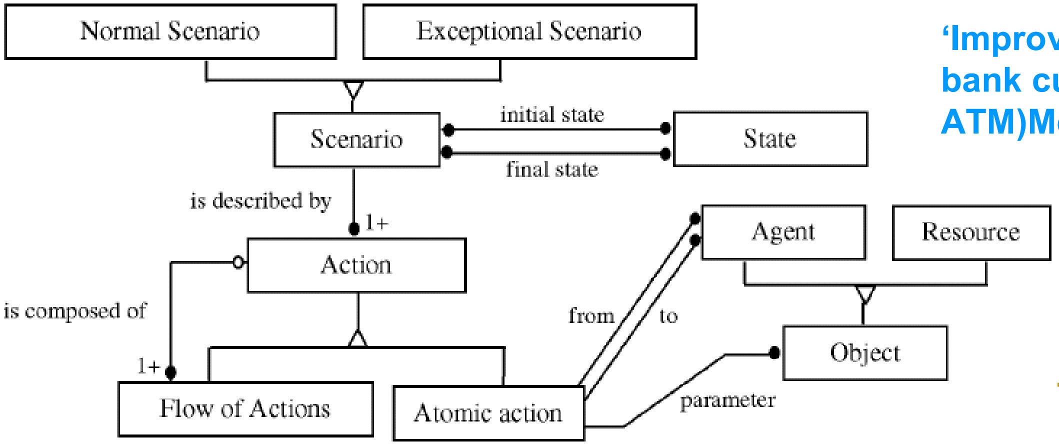
‘Take (the receipt)Obj (from the printer)So’;

‘Read (the validity date of card)Obj (in the card chip)So’;

‘Display (the error message)Obj (to the customer)Dest’;

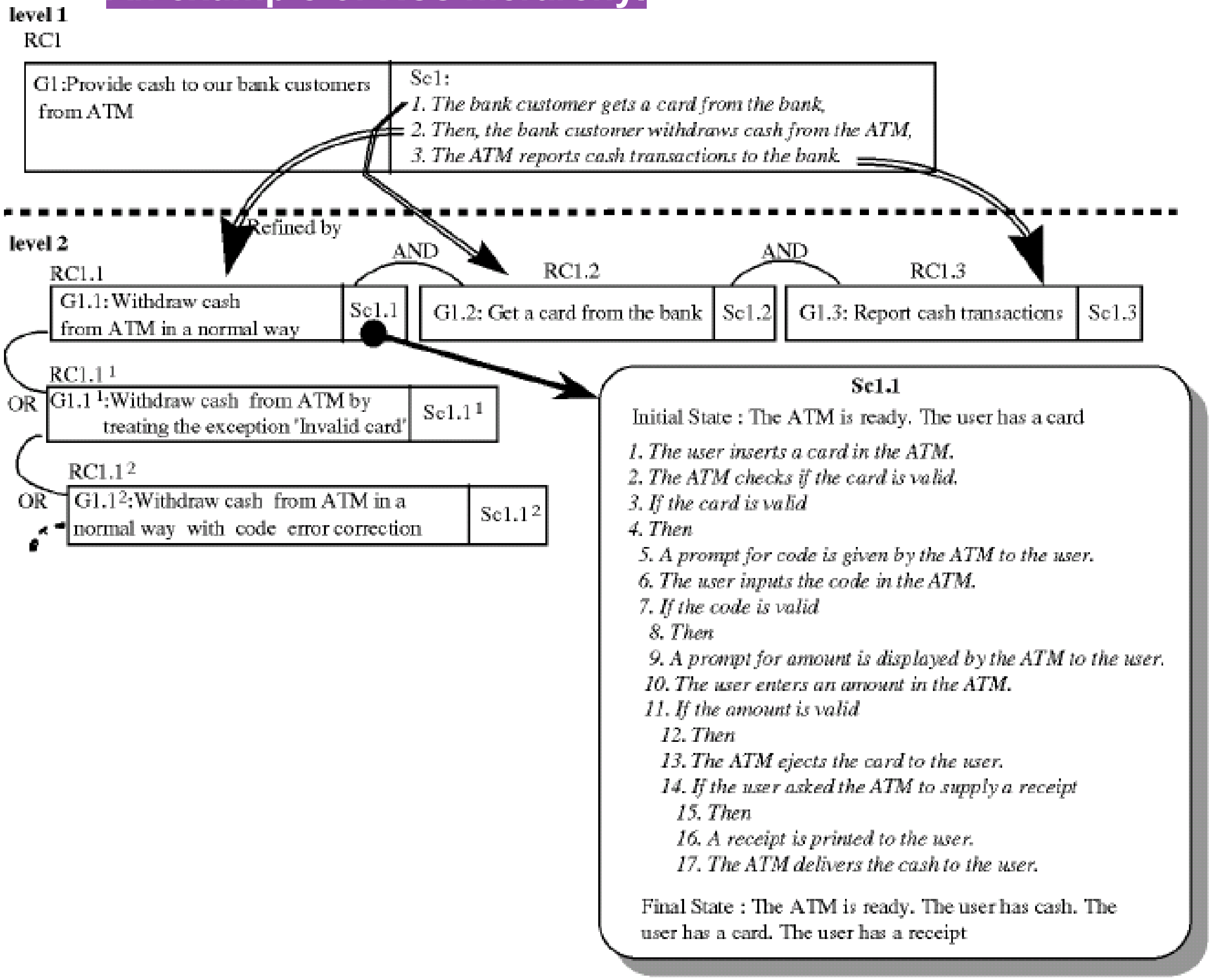
‘Improve (our services)Obj (by providing (cash)Obj (to our bank customers)Dest (from account)So(with a card based ATM)Mea)Man’,

Scenario structure



Goals & Scenarios

An example of RCs hierarchy.



level 1

RC1

G1:Provide cash to our bank customers from ATM

Sc1:
1. The bank customer gets a card from the bank,
2. Then, the bank customer withdraws cash from the ATM,
3. The ATM reports cash transactions to the bank.

Refined by

level 2

AND

AND

RC1.1

G1.1: Withdraw cash from ATM in a normal way | Sc1.1

RC1.2

G1.2: Get a card from the bank | Sc1.2

RC1.3

G1.3: Report cash transactions | Sc1.3

OR

RC1.1 1

G1.1 1:Withdraw cash from ATM by treating the exception 'Invalid card' | Sc1.1 1

OR

RC1.1 2

G1.1 2:Withdraw cash from ATM in a normal way with code error correction | Sc1.1 2

Sc1.1

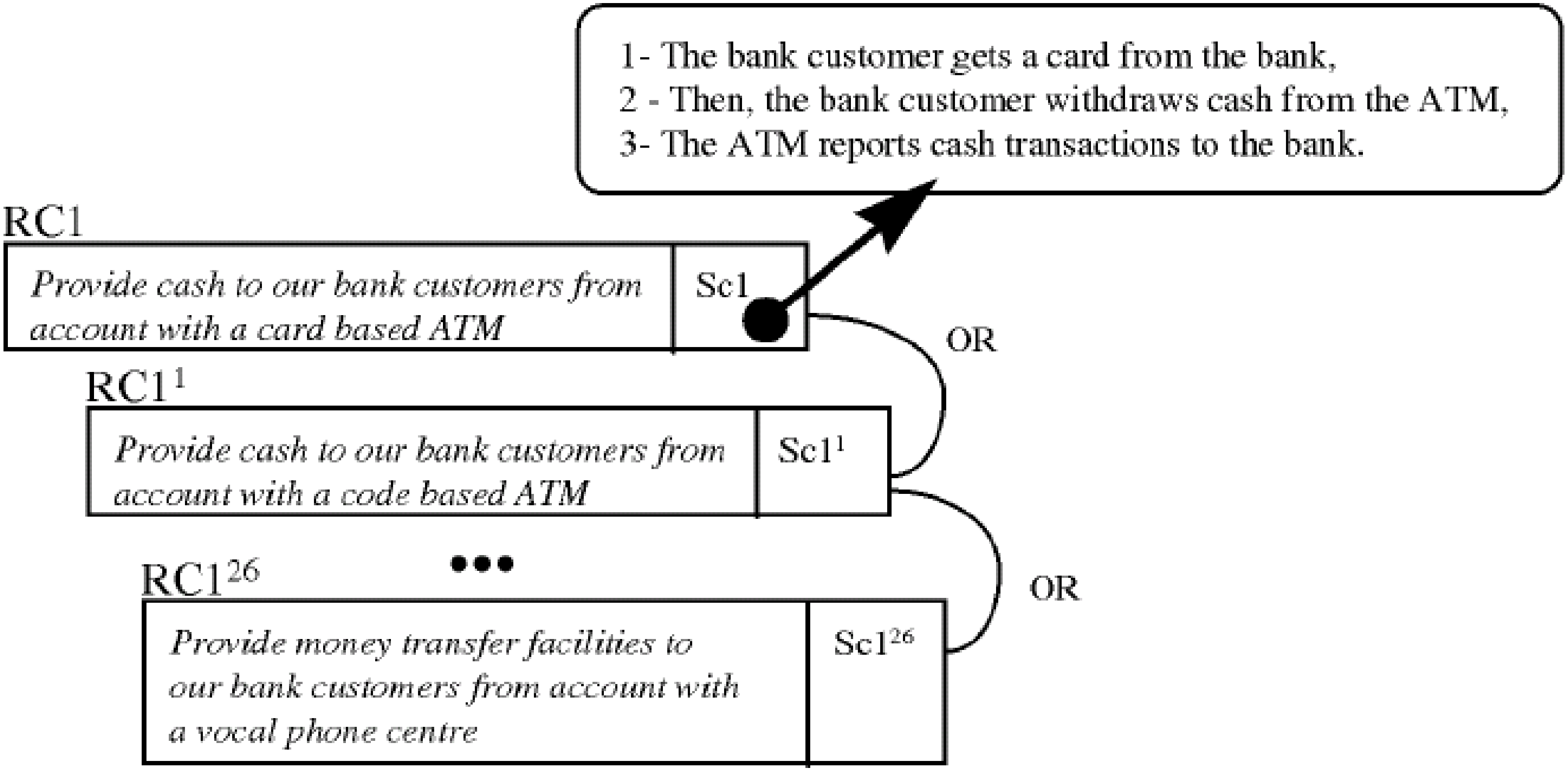
Initial State : The ATM is ready. The user has a card

1. The user inserts a card in the ATM.
2. The ATM checks if the card is valid.
3. If the card is valid
4. Then
5. A prompt for code is given by the ATM to the user.
6. The user inputs the code in the ATM.
7. If the code is valid
8. Then
9. A prompt for amount is displayed by the ATM to the user.
10. The user enters an amount in the ATM.
11. If the amount is valid
12. Then
13. The ATM ejects the card to the user.
14. If the user asked the ATM to supply a receipt
15. Then
16. A receipt is printed to the user.
17. The ATM delivers the cash to the user.

Final State : The ATM is ready. The user has cash. The user has a card. The user has a receipt

Goals & Scenarios

Contextual chunks discovered from the contextual chunk RC1.



1- The bank customer gets a card from the bank,
2 - Then, the bank customer withdraws cash from the ATM,
3- The ATM reports cash transactions to the bank.

RC1

Provide cash to our bank customers from account with a card based ATM — Sc1

$RC1^{1}$

Provide cash to our bank customers from account with a code based ATM — $Sc1^{1}$

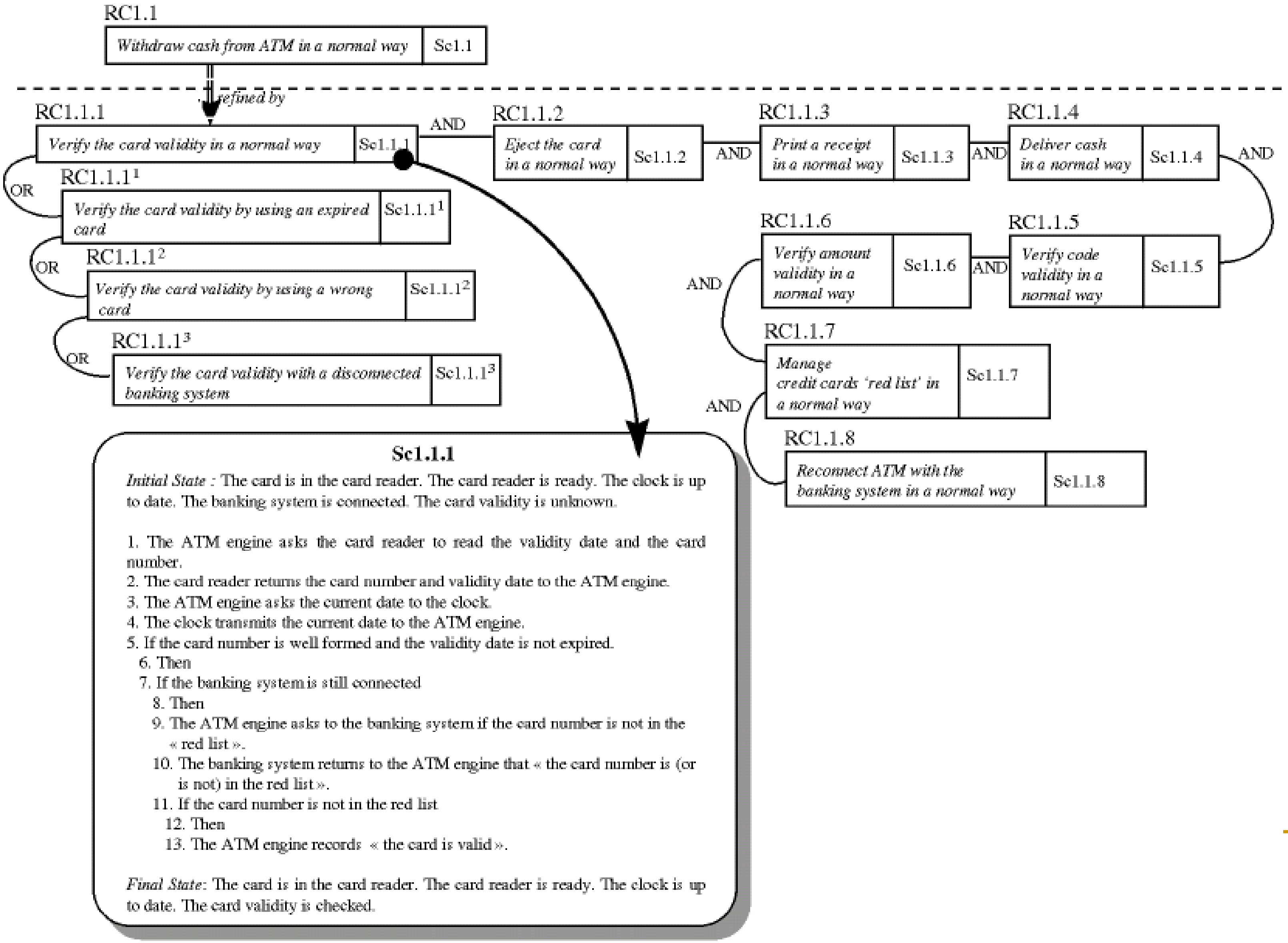
•••

$RC1^{26}$

Provide money transfer facilities to our bank customers from account with a vocal phone centre — $Sc1^{26}$

Goals & Scenarios

System internal chunks discovered from the system interaction chunk RC1.1.



RC1.1
Withdraw cash from ATM in a normal way Sc1.1

refined by

RC1.1.1
Verify the card validity in a normal way Sc1.1.1

OR

RC1.1.1¹
Verify the card validity by using an expired card Sc1.1.1¹

OR

RC1.1.1²
Verify the card validity by using a wrong card Sc1.1.1²

OR

RC1.1.1³
Verify the card validity with a disconnected banking system Sc1.1.1³

AND

RC1.1.2
Eject the card in a normal way Sc1.1.2

AND

RC1.1.3
Print a receipt in a normal way Sc1.1.3

AND

RC1.1.4
Deliver cash in a normal way Sc1.1.4

AND

RC1.1.5
Verify code validity in a normal way Sc1.1.5

AND

RC1.1.6
Verify amount validity in a normal way Sc1.1.6

AND

RC1.1.7
Manage credit cards 'red list' in a normal way Sc1.1.7

AND

RC1.1.8
Reconnect ATM with the banking system in a normal way Sc1.1.8

Sc1.1.1

Initial State : The card is in the card reader. The card reader is ready. The clock is up to date. The banking system is connected. The card validity is unknown.

1. The ATM engine asks the card reader to read the validity date and the card number.
2. The card reader returns the card number and validity date to the ATM engine.
3. The ATM engine asks the current date to the clock.
4. The clock transmits the current date to the ATM engine.
5. If the card number is well formed and the validity date is not expired.
   6. Then
   7. If the banking system is still connected
      8. Then
      9. The ATM engine asks to the banking system if the card number is not in the « red list ».
      10. The banking system returns to the ATM engine that « the card number is (or is not) in the red list ».
      11. If the card number is not in the red list
         12. Then
         13. The ATM engine records « the card is valid ».

Final State: The card is in the card reader. The card reader is ready. The clock is up to date. The card validity is checked.

Misuse Cases

[G. Sindre, A. L. Opdahl: Eliciting Security Requirements by Misuse Cases. TOOLS (37) 2000: 120-131]

Compromised Use Case Properties

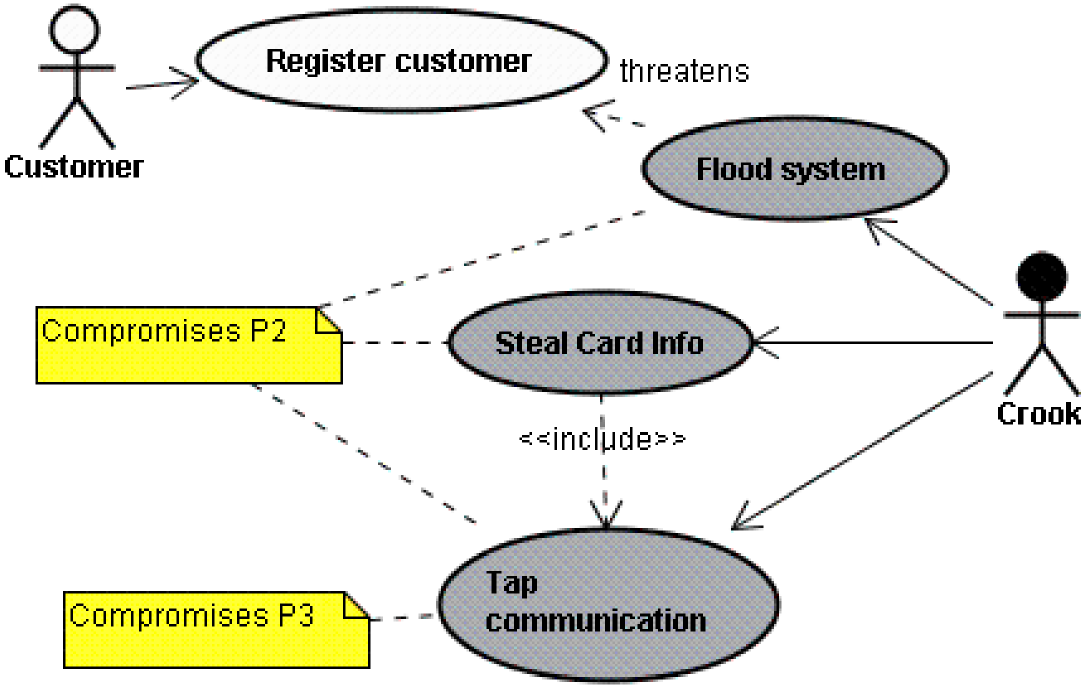
P1 – System boundary identifies the scope of the subject (the target system).

P2 – Use case describes system actions or actor-system interactions.

P3 – Use case describes actions from black-box perspective

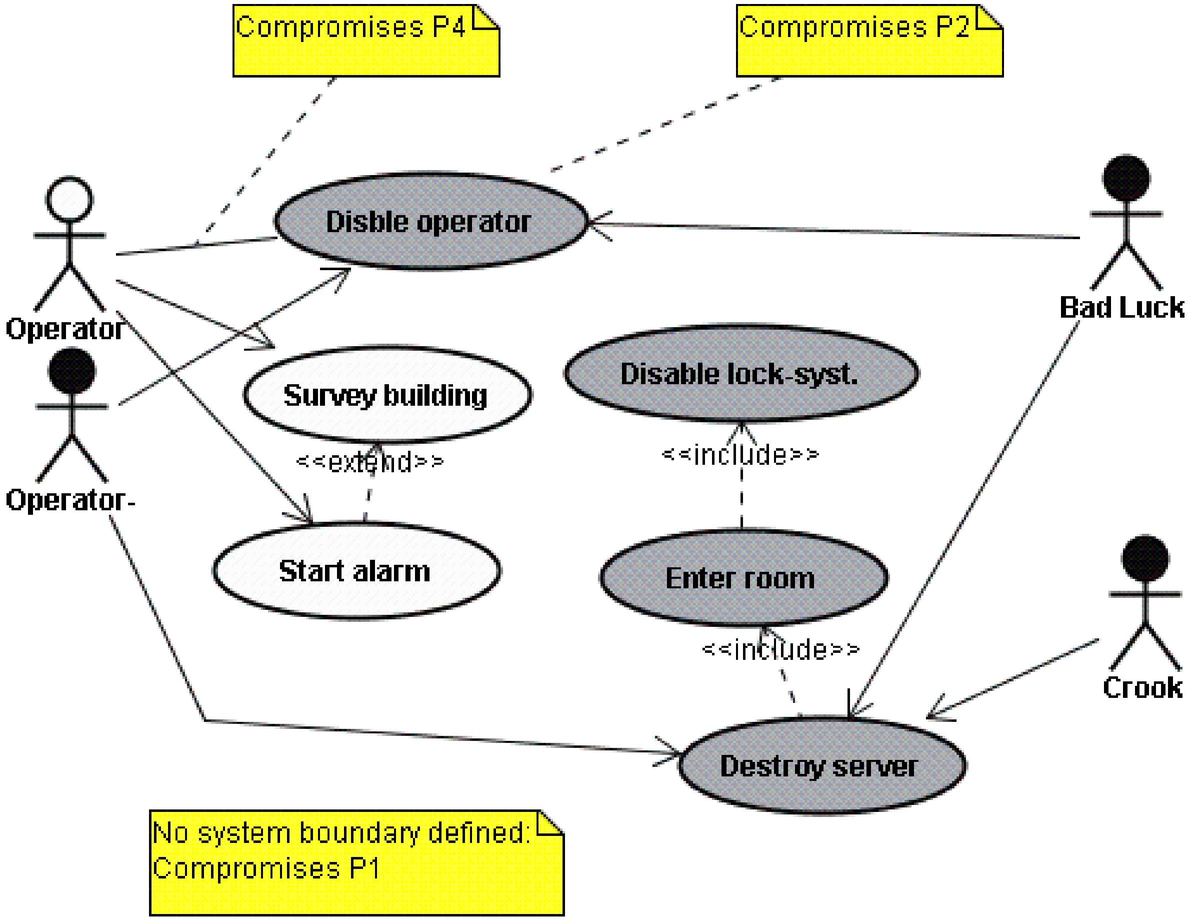
P4 – Communicate Association represents communication between actor and use case

Example 1



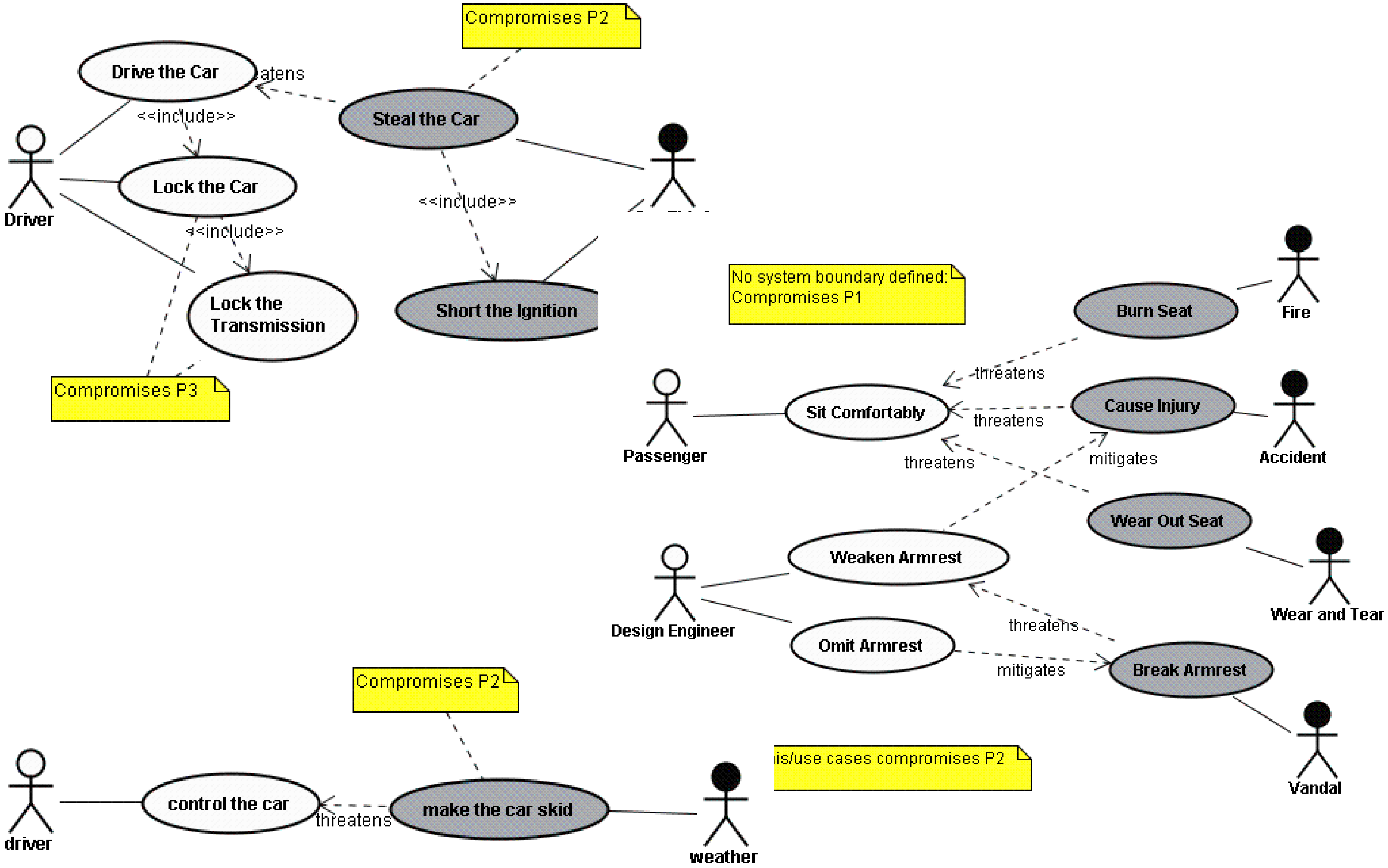
Misuse Cases

Example 2



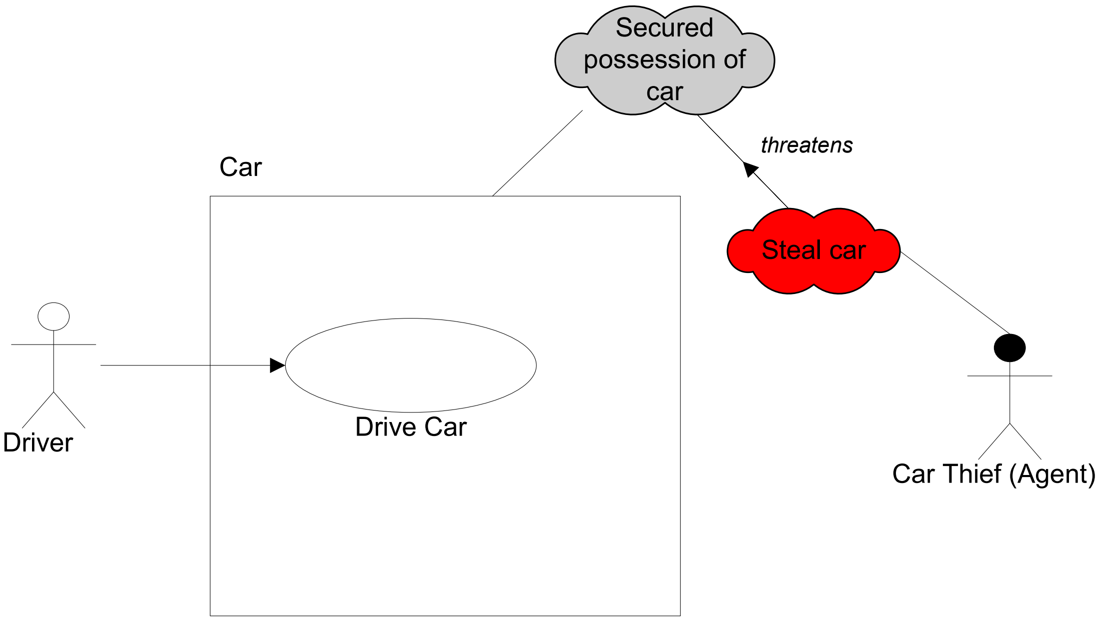
Misuse Cases

[Ian F. Alexander: Initial Industrial Experience of Misuse Cases in Trade-Off Analysis. RE 2002: 61-70]



Misuse Cases or Anti-goals

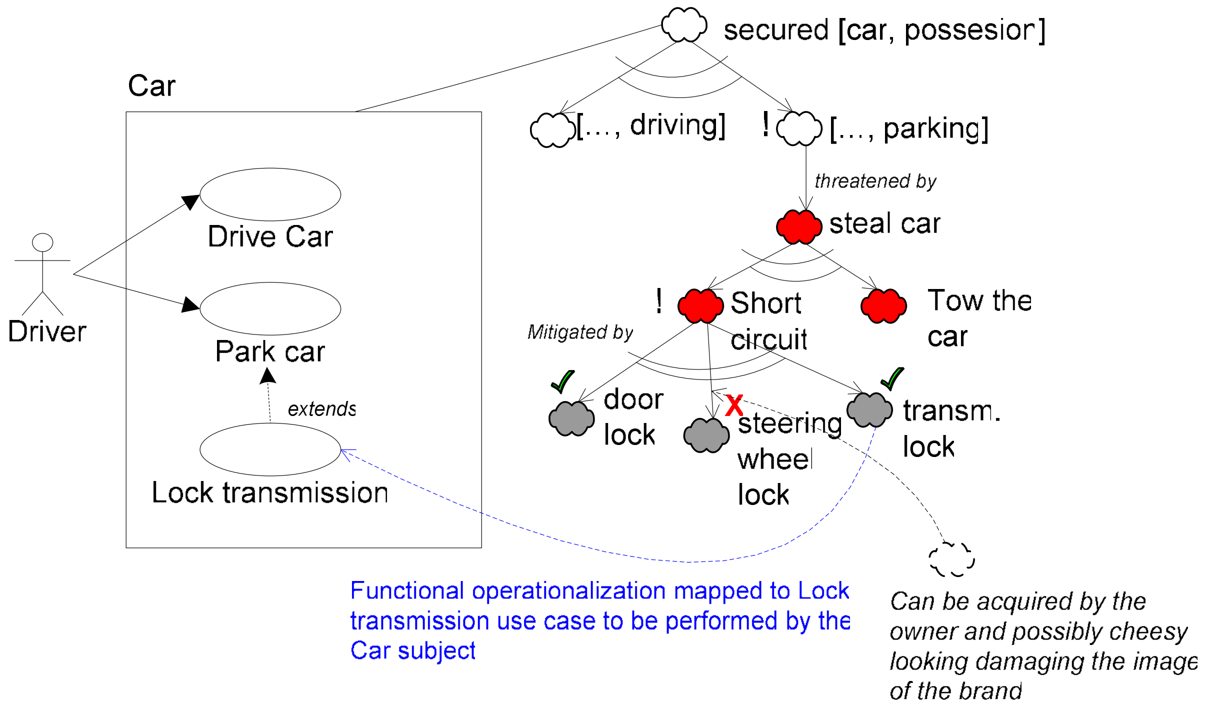
[S. Supakkul and L. Chung, A Goal-Oriented Misuse Analysis, a working memo;
Axel van Lamsweerde: Elaborating Security Requirements by Construction of Intentional Anti-Models. ICSE 2004: 148-157]



Step 1. Identify “Steal car” as an anti-goal. Anti-goals allows us to represent any undesirable effect that are not limited to system actions as with misuse cases.

Step 2. Identify “Steal car” as an NFR anti-softgoal, “Secured possession of car” as an NFR softgoal, and “threatens” as the a contribution between the two.

Misuse Cases vs. Anti-goals



Step 3. Refine NFR softgoals to determine that Topic [car, possession] can be decomposed to [car, possession, while driving] and [car, possession, while parking]. Assign an exclamation mark (!) to indicate that secured possession while parking is more critical.

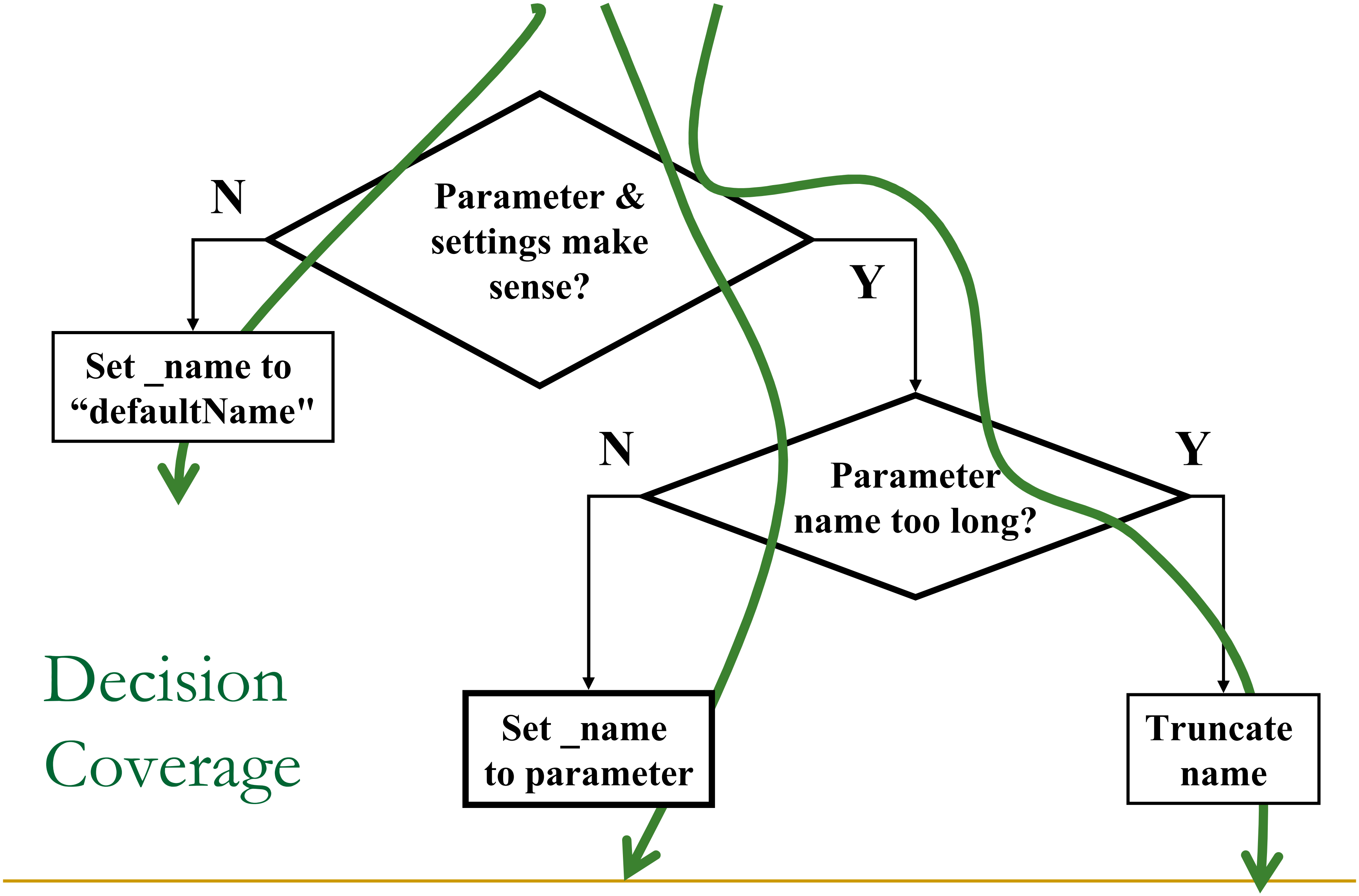
Step 4. Re-associate the anti-goal “steal car” to “secured[car, possession, while parking] to provide a better context that we’re concerned with only car begin stolen while parking.

Step 5. Refine car stealing technique to “short circuit” and “tow the car” and assign an “!” to indicate that “short circuit” technique is what we’re concerned with.

Step 6. Identify operationalization to mitigate “short circuit” using “door lock”, “steering wheel lock”, and “transmission lock”. Determine that door lock and trans would be provided by the car.

Step 7. Map trans. Lock to a use case extending the newly defined “Park car” use case.

Paths to be Checked



Decision Coverage

Adapted from *Software Engineering: An Object-Oriented Perspective* by Eric J. Braude (Wiley 2001), with permission.