Getting Started

We now have dress shirts on sale for men with 16 necks
[at a department store]

Ice cream souveniers *[on a billboard on I-24, Nashville]*

Thou shall commit adultery *[The Wicked Bible, 1623, England]*

Slow children at play *[in residential areas]*

Please wait for hostess to be seated *[at restaurants]*

We will sell gasoline to anyone in a glass container
[Santa Fe gas station]

Don't kill your wife. *[In the window of a Kentucky appliance store]*
Let our washing machine do the dirty work.

Dinner Special - Turkey $2.35; Chicken or Beef $2.25;
Children $2.00

Will the last person to leave please see that the perpetual light is extinguished *[In the vestry of a New England church]*

When two trains approach each other at a crossing, both shall come to a full stop and neither shall start up again until the other is gone. *[Kansas legislature, early 1890's]*

Trespassers will be prosecuted to the full extent of the law
- Sisters of Mercy *[on the wall of a Baltimore estate]*

Man, honest. Will take anything *[an ad]*

Requirements Elicitation

- Why is it difficult?
  - Critical Issues
- What to elicit?
  - Four Worlds of RE
  - Models about Requirements Revisited
- How to elicit?
  - Desirable Properties Revisited
  - Goal-Oriented Elicitation
    - Classical Logic Approach
      - Traceability
      - Approaches With A Richer Ontology
  - Domain Analysis
  - Problem Frames
  - Data/Information Elicitation Techniques

Refresher

What are RE Processes?

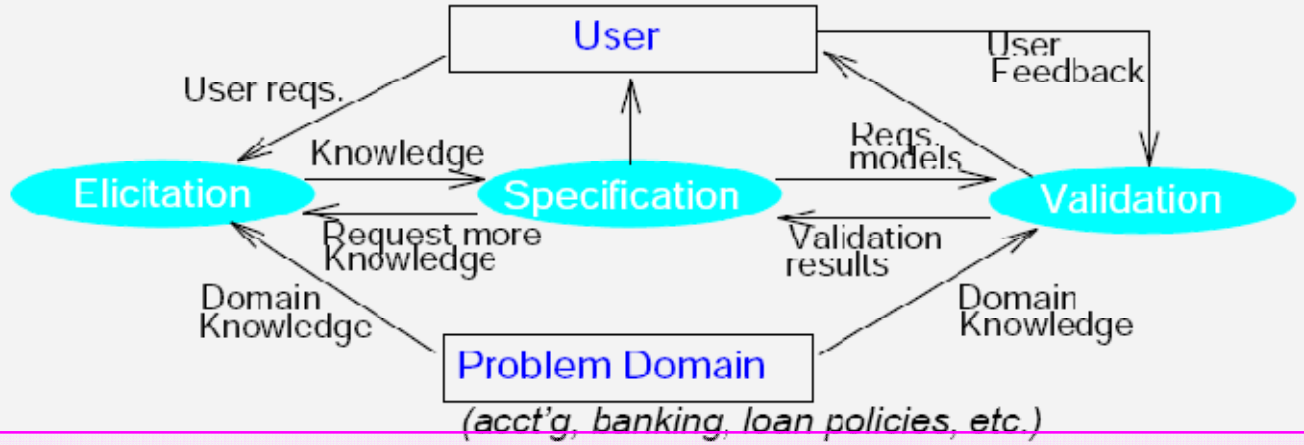
A Framework

for intial model construction & subsequent reengineering

3 fundamental concerns: *understand* the problem

(formally) describe

attain an agreement on



- **Elicitation**
  determine what's really needed, why needed, whom to talk to
  acquire as much knowledge as possible

- **Specification**
  produce a (formal) RS model: translate "vague" into "concrete", etc.
  make various decisions on what & how

- **Validation**
  assure that the RS model satisfies the users' needs

early RE

late RE

Refresher

Why is it difficult?

A wicked problem

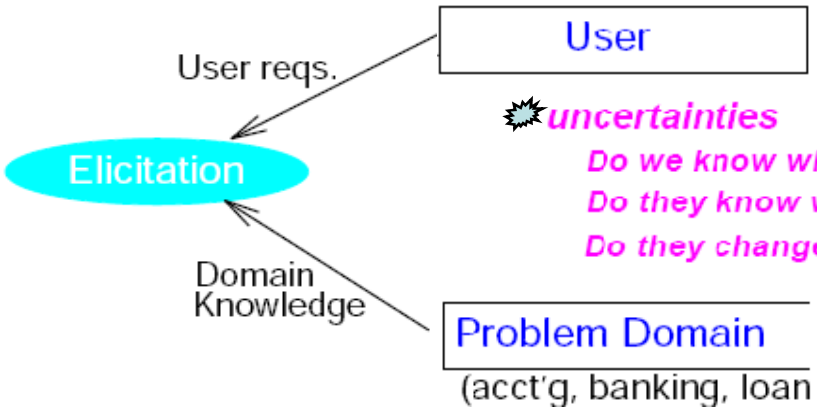
- *many sources*
  - *customers, tellers, other employees*
  - *databases, files, chapters, literature, standards, legal issues*
  - *surveys of similar applications*
    - *(what are they, who are they, how successful are they)*
- *different views on wants and needs*
- *different notations, mental models*

Relativism is everywhere:

Truth often depends on the observer

- The meanings of the words
- The assumptions about context
- The set of categories for understanding the world
- perceptual limitations
- cognitive ability
- personal values and experience

"In the land of the blind, the one-eyed man is king"



- *uncertainties*
  - *Do we know whom to talk to?*
  - *Do they know what they want/need?*
  - *Do they change their minds?*
- *many sources*
  - *domain experts (accountants, bankers, loan managers, etc.)*
  - *databases, files, chapters, literature, standards, legal issues*

Refresher

Why is it difficult?

A wicked problem

- identification process complex (repetitive interactions)
- communication, coordination process complex

- requirements volatility:

  Reqs. change because the problem being solved changes,
  because people's perception changes,
  because some involved persons were not contacted or
  were contacted but not in an appropriate manner.

Requirements creeping rate = percentage of change/time

Can you cope with 50%/month?

Ethnomethodology

Why is it difficult?

A wicked problem

the "say-do" problem: people know how to do things they normally don't describe (tacit knowledge); descriptions of such things may be highly inaccurate

So, what should be done then?

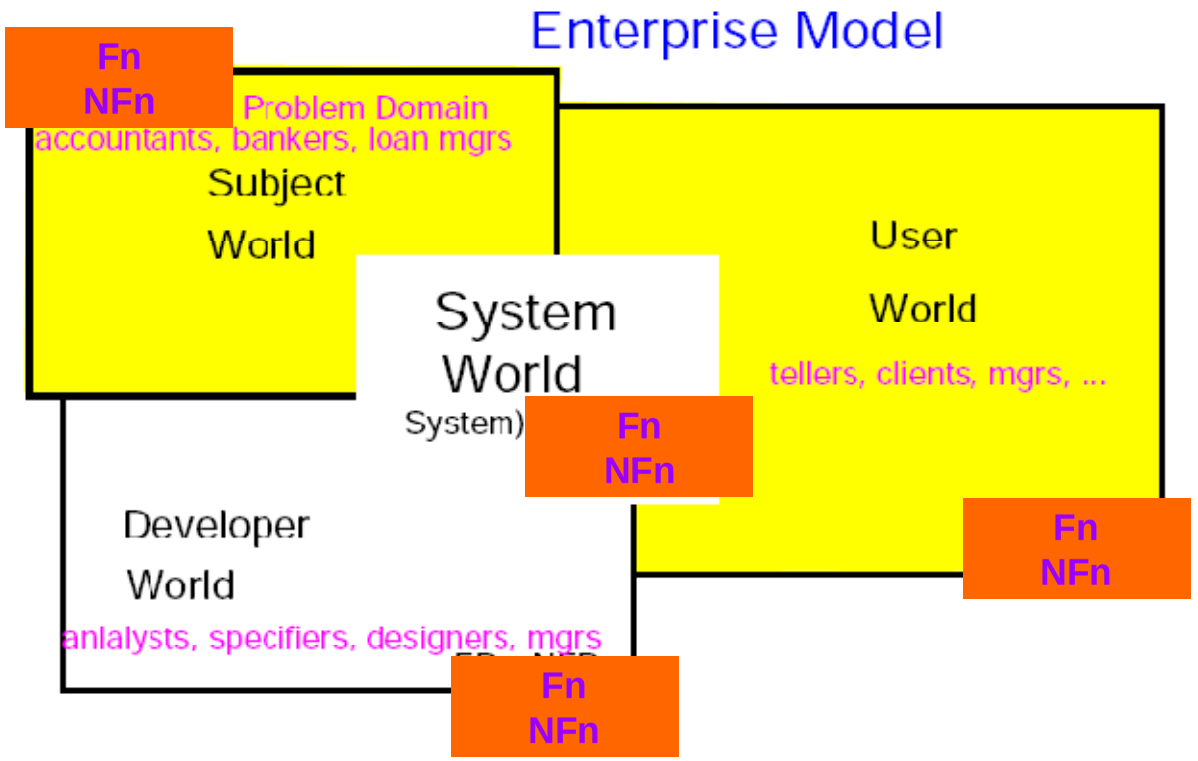
List three examples

Requirements Elicitation

- Why is it difficult?
  - Critical Issues
- What to elicit?
  - Four Worlds of RE
  - Models about Requirements Revisited
- How to elicit?
  - Desirable Properties Revisited
  - Goal-Oriented Elicitation
    - Classical Logic Approach
      - Traceability
      - Approaches With A Richer Ontology
  - Domain Analysis
  - Problem Frames
  - Data/Information Elicitation Techniques

What to elicit?

Four Worlds of RE



Does the reference model capture all the above?

Where are goals, services and constraints? [Zave94]

Which is about S, D |= R?

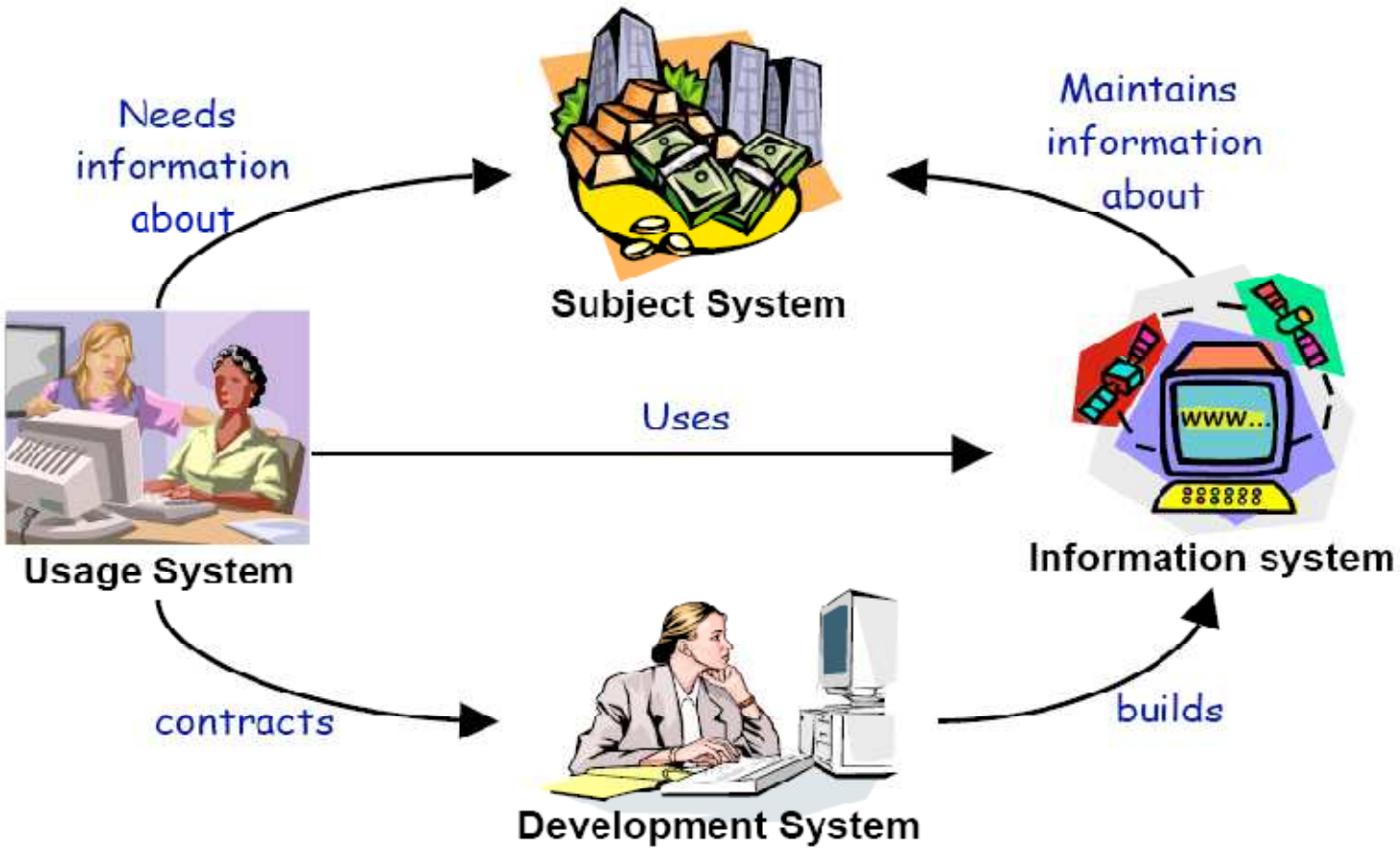
Which is about technical feasibility, component reuse, etc.?

Where is traceability?

Requirements should contain *nothing but* information about the environment.

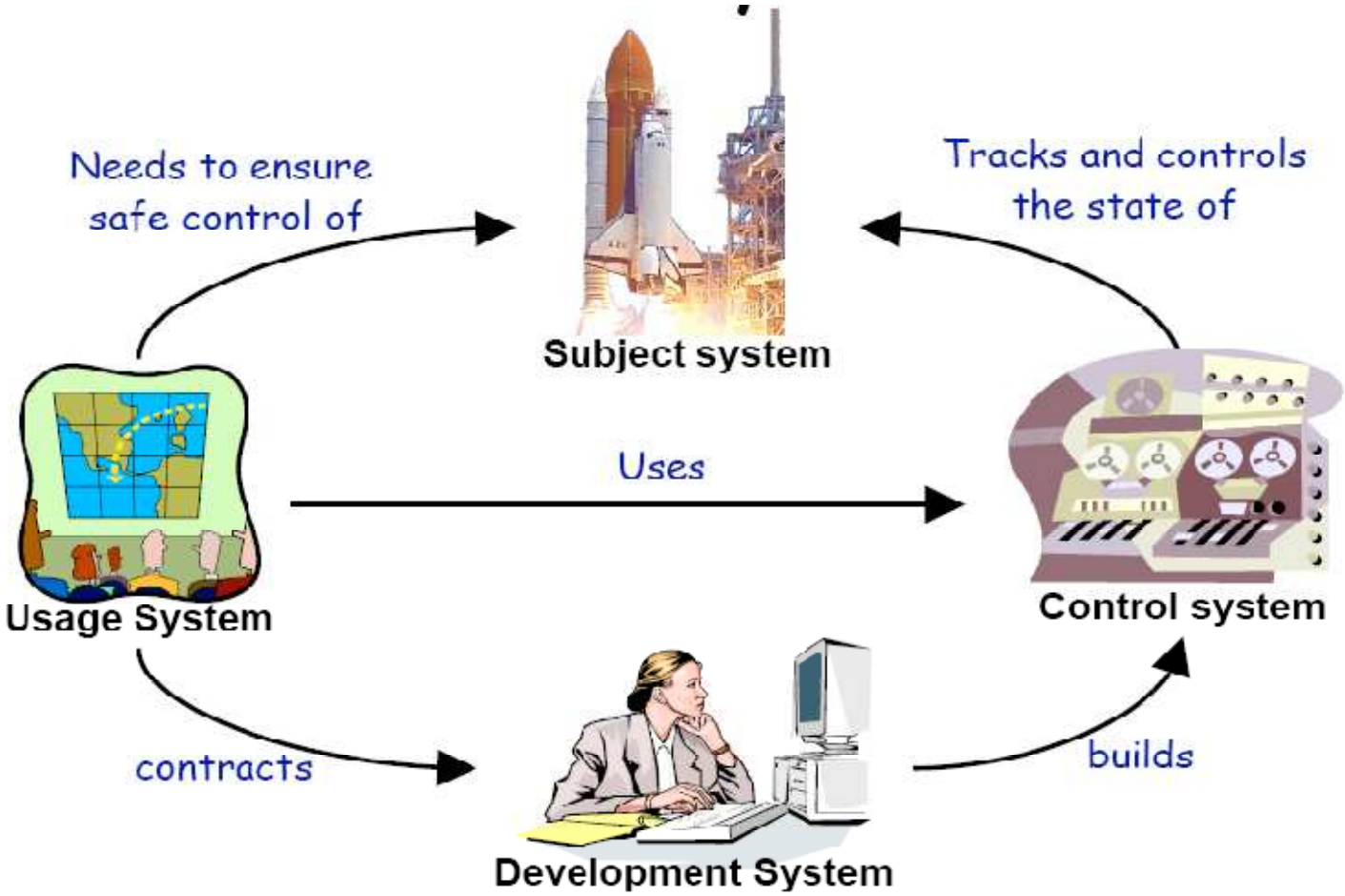
Four Worlds of RE for Information Systems

Adapted from [LK1995, p73] [S. Easterbrook, 2000-2004]



How does relate to RE process?

Four Worlds of RE for Control Systems

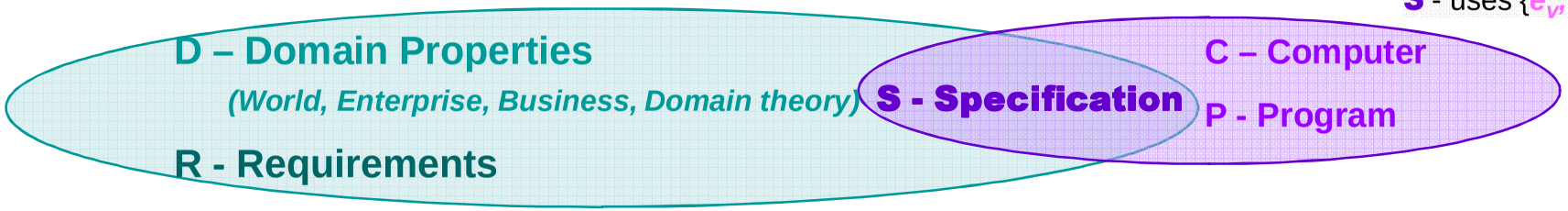


Recall: Models about Requirements Revisited

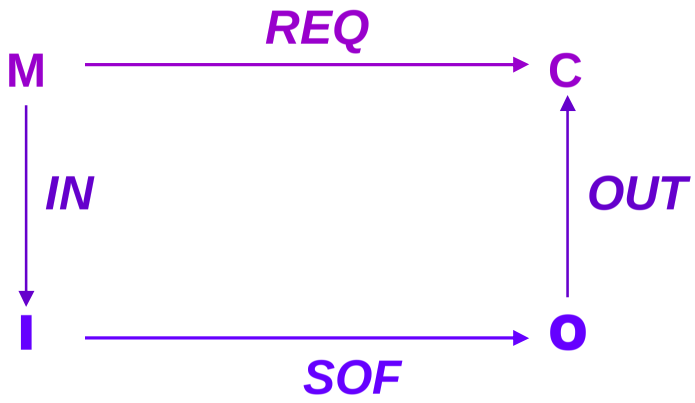
The Why-What-How Model

The WRSPM Model

W, **R** - uses {e_h, e_v, s_v}
P, **M** - uses {e_v, s_v, s_h}
S - uses {e_v, s_v}



- ***The 4-variable model:***



- ***The goal-service-constraint model:***

Requirements Elicitation

- Why is it difficult?
  - Critical Issues
- What to elicit?
  - Four Worlds of RE
  - The Reference Model Revisited
- How to elicit?
  - Desirable Properties Revisited
  - Goal-Oriented Elicitation
    - Classical Logic Approach
      - Traceability
    - Approaches With A Richer Ontology
  - Domain Analysis
  - Problem Frames
  - Data/Information Elicitation Techniques

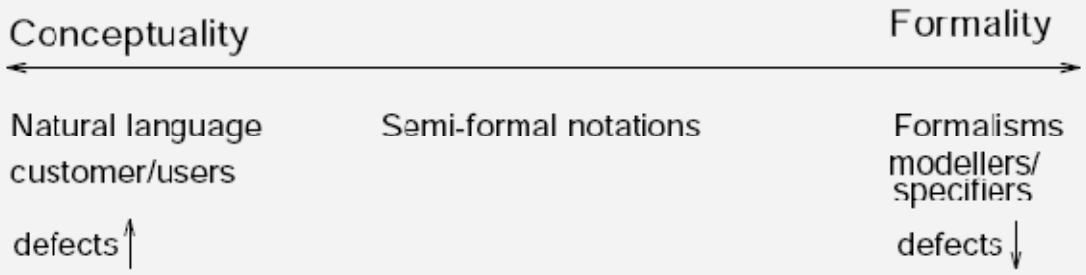
How to elicit?

Recall

How to do RE?

Major themes of the course

- **Error detection and removal**



- Clarity

shall, will, must, should, etc. => formailty (criticality)

A or B => formality (inclusive-or/exclusive-or)

mostly A => formality (A in cases C1, C2, ..., Cn)

perhaps/could/may/might A => formality (A in cases C1, C2, ..., Cn)
by and large, often, frequently A => formality (A in cases C1, C2, ..., Cn)

A or ~A => formally ignore as tautology, but may mean something

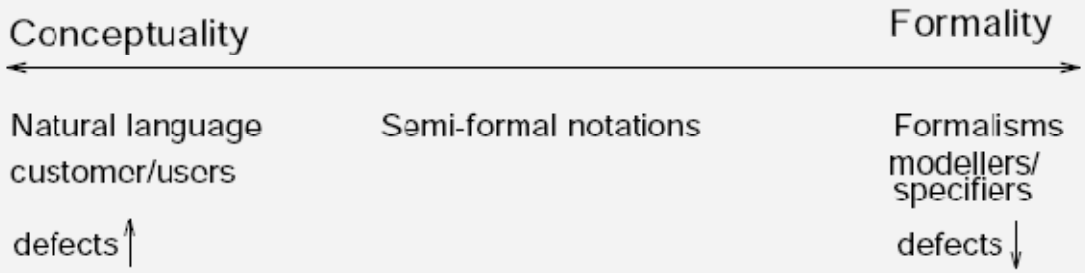
How to elicit?

Recall

How to do RE?

Major themes of the course

Error detection and removal



Consistency

A set of formulas Φ in first-order logic is **consistent** if and only if there is no formula φ such that $\Phi \vdash \phi$ and $\Phi \vdash \neg\phi$.

A and ~A => formally false, ignorable, but could be typo

A implies B A ~B	modus ponens

A or B ~A ~B

Slow children at play => semantic issue

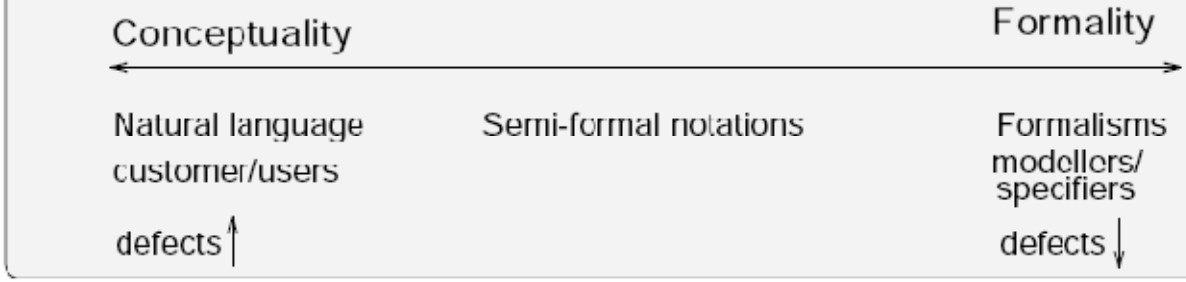
How to elicit?

Recall

How to do RE?

Major themes of the course

Error detection and removal



Correctness (external Consistency)

When two trains approach each other at a crossing, both shall come to a full stop and neither shall start up again until the other is gone. [Kansas legislature, early 1890's]

=> scenario analysis and/or
=> formalism

Call forwarding: (B -> C); (A -> C); (D -> A)
(B -> C); (C -> B)

How would you detect these potential problems?
How should you deal with "feature interaction problems"? [Zave]

How to elicit?

Recall

How to do RE?

Major themes of the course

Error detection and removal

Conceptuality ⟷ Formality

Natural language	Semi-formal notations	Formalisms
customer/users		modellers/ specifiers
defects ↑		defects ↓

Completeness *sometimes, "the" key issue*

hard => elicitation techniques

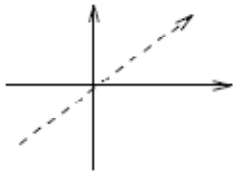
(logical formalism is modern philosophy)

=> use of "ontological" primitives

goals, agents, decisions, rationale ***goal-orientation, agent-orientation***

entities, activities, constraints ***object-orientation***

=> use of organizational primitives



classes/metaclasses,
associations/aggregations,
superclasses/subclasses
views

In philosophy, ontology is the study of being or existence. It seeks to describe or posit the basic categories and relationships of being or existence to define entities and types of entities within its framework. Ontology can be said to study conceptions of reality. http://en.wikipedia.org/wiki/Ontology

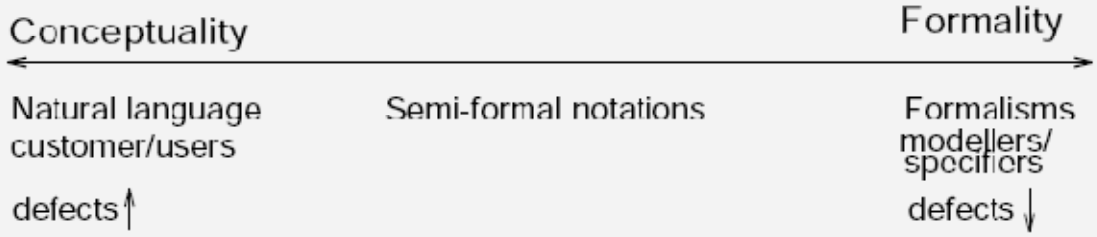
How to elicit?

Recall

How to do RE?

Major themes of the course

Error detection and removal



Overspecification

algorithms (sorting, searching, routing, serialization, normalization, etc.)
data structures (stack, queue, tree, graph, heap, etc.)

Technically unsolvable problems

scheduling, pattern recognition, etc.

Also, do not worry at this time about acquiring the resources to build the house. Your first priority is to develop detailed plans and specifications. Once I approve these plans, however, I would expect the house to be under roof within 48 hours.

While you are designing this house specifically for me, keep in mind that sooner or later I will have to sell it to someone else. It therefore should have appeal to a wide variety of potential buyers. Please make sure before you finalize the plans that there is a consensus of the population in my area that they like the features this house has.

I advise you to run up and look at my neighbor's house he constructed last year. We like it a great deal. It has many features that we would also like in our new home, particularly the 75-foot swimming pool. With careful engineering, I believe that you can design this into our new house without impacting the final cost.

Lawrence Chung

Recall

Another Look at Completeness $S, K \vdash R$

[p. Zave and M. Jackson, Four Dark Corners of Requirements Engineering. *ACM Transactions on Software Engineering and Methodology* 6(1) 1-30. ACM Press. 1997.

If the five following criteria are satisfied, then requirements engineering, in the strongest sense, is complete.

(1) There is a set R of requirements. Each member of R has been validated (checked informally) as acceptable to the customer, and R as a whole has been validated as expressing all the customer's desires with respect to the software-development project.

(2) There is a set K of statements of domain knowledge. Each member of K has been validated (checked informally) as true of the environment.

(3) There is a set S of specifications. The members of S do not constrain the environment, they are not stated in terms of any unshared actions or state components, and they do not refer to the future.

(4) A proof shows that:

$$S, K \vdash R$$

This proof ensures that an implementation of S will satisfy the requirements.

$$P, C \vdash S$$

If the machine is as described in C, then execution of the program P will ensure the behaviour S at the machine's interface with the world.

(5) There is a proof that S and K are consistent. This ensures that the specification is internally consistent and consistent with the environment. Note that the two proofs together imply that S, K, and R are consistent with each other.

… requirements should contain *nothing but* information about the environment. [p9]

The requirements are complete if they are sufficient to establish the goal they are refining

[K. Yue, "What Does It Mean to Say that a Specification is Complete?", Proc. IWSSD-4, Fourth International Workshop on Software Specification and Design, Monterey, 1987.]

Refresher

How to Elicit?
Goal-oriented Requirements Elicitation

"... Requirements Engineering is the branch of Systems engineering concerned with

real-world goals for,
services provided by, and
constraints on
software systems

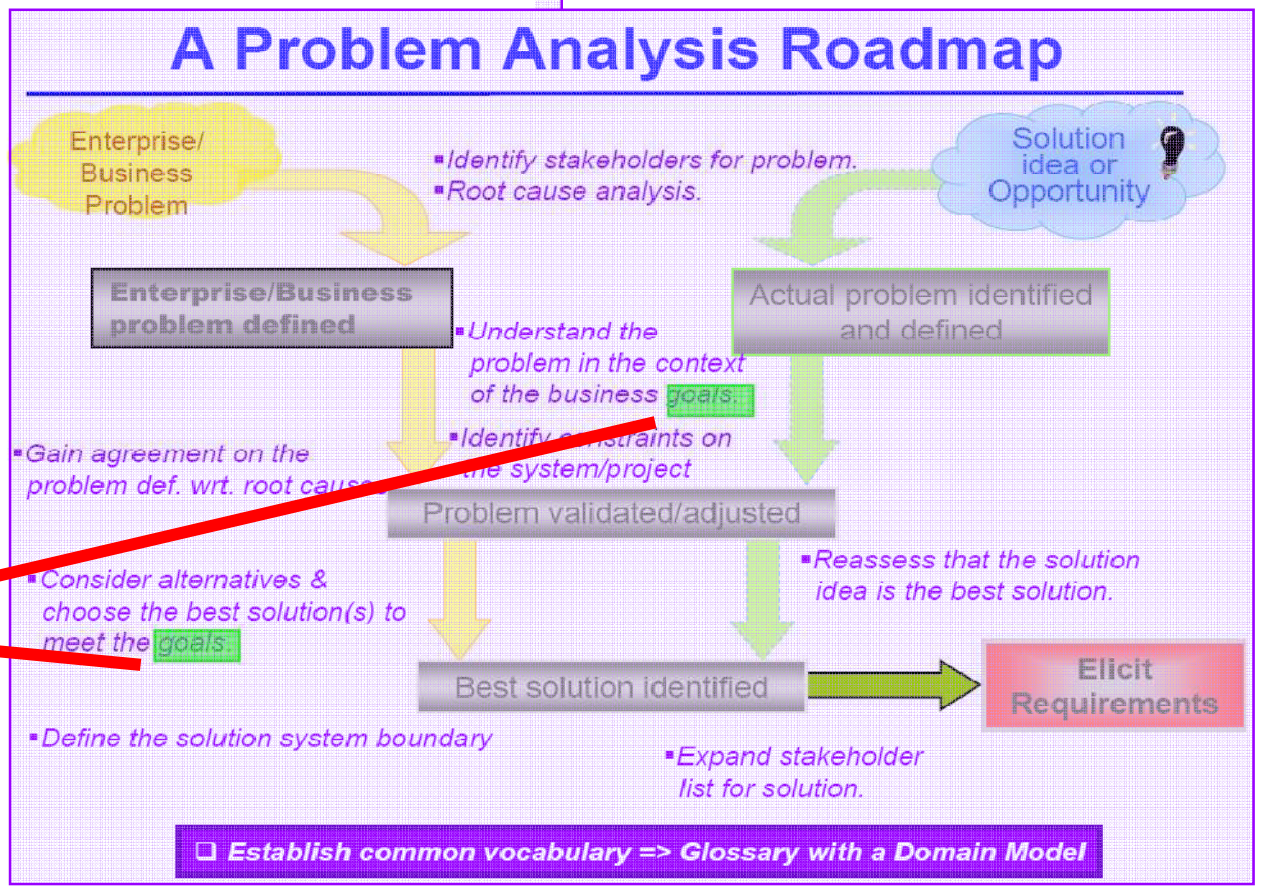
Requirements engineering is also

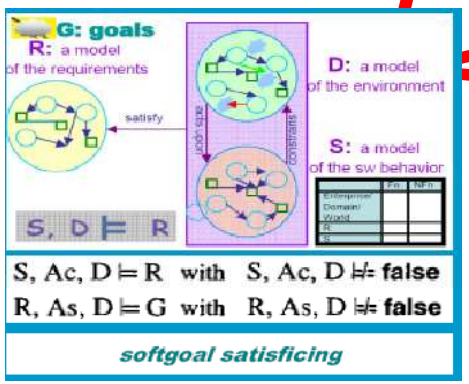
the relationships of these fa

to precise specifications
and
to their evolution over ti

Where is Use Case Diagram?

A Problem Analysis Roadmap





How to elicit?

Teleological view of systems

tel-e-ol-o-gy n., pl. -gies. 1. The philosophical study of design or purpose in natural phenomena. 2. The use of ultimate purpose or design as a means of explaining natural phenomena.
(webster)

A system has a set of goals it seeks to attain;
a system's behavior is explained in terms of its goals.

-> traceability, justifiability

- E.g., Library System

send an overdue notice via mail
- why? it takes time and money
  - via email/phone
  - within library use only
  - send an early notice

Alternatives

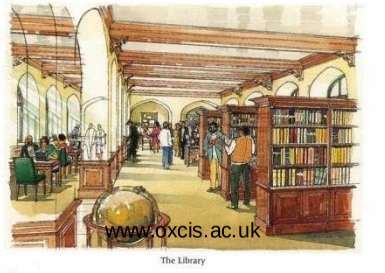
to ensure books are regularly available
- what's regular availability?

books in shelf
- why?

to satisfy book request
- why?

to make the system effective

high-level goal



How to elicit?

Teleological view of systems -> Goal-Directed Strategy

interleaving, iterative

post a goal
refine the goal
do everything to meet the goal
  consider alternatives
  analyze tradeoffs

Elicitation

User

Problem Domain

Goal-Directed Strategy -> Goal structure

Goals initially stated by the client are incrementally refined into subgoals

traditionally AND/OR decompositions

to satisfy the parent,
satisfy all its descendants

to satisfy the parent,
satisfy any of its descendants

How to elicit?

Goal-Directed Strategy 1: Classical Logic Approach

Expressive Power

"Propositional and predicate logic provide all the basic concepts needed for a systematic engineering design methodology"

[C. A. Hoare, Mathematical Models for Computing Science, Oxford Univ. Rpt., Aug. 1994]

Example: "good old vending machine"

step 1: identify the top goal ---> "serve_customer"
identify more domain-specific goals ---> "dispensing cash", "serving coffee", "vending candy bars", "shining shoes", etc.

=> domain analysis

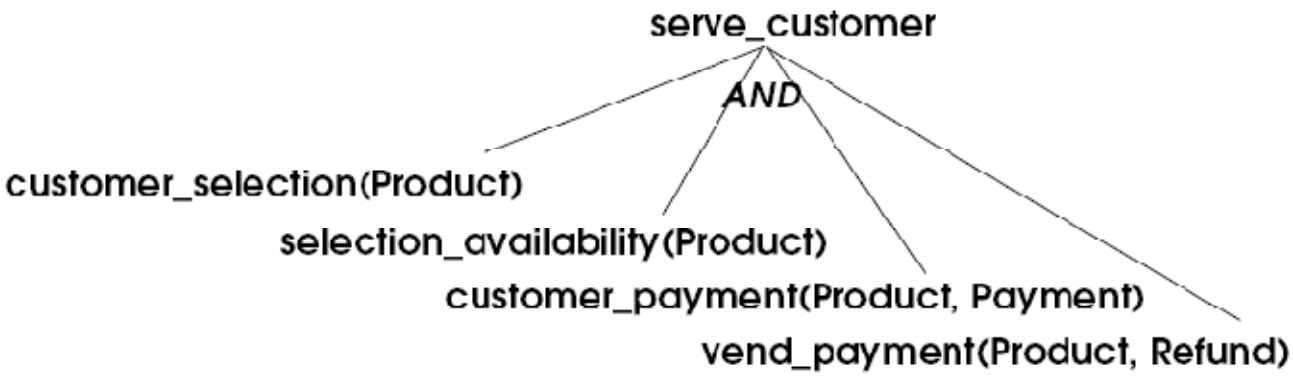
step 2: examine what is (really) needed to satisfy the goal.

- what product the customer wants
- if the vending machine can dispense the customer's choice (has sufficient inventory?)
- if the customer has the resources (cash or credit card) and is ready to pay for the selection
- if the customer has deposited more money than necessary for the purchase
- if the selection and any change were properly dispensed.

How to elicit?

Goal-Directed Strategy 1: Classical Logic Approach

- Example: "good old vending machine"



- ---> Executable specification (e.g., in Prolog)

```
serve_customer :- customer_selection(Product),
                  selection_availability(Product),
                  customer_payment(Product, Payment),
                  vend_payment(Product, Refund).
```

H :- B1, …, Bn. *to show/solve H, show/solve B1 and … and Bn.*

e.g.,

```
sibling(X,Y) :- parent(Z,X), parent(Z,Y).
parent(X,Y) :- father(X,Y).
parent(X,Y) :- mother(X,Y).
```

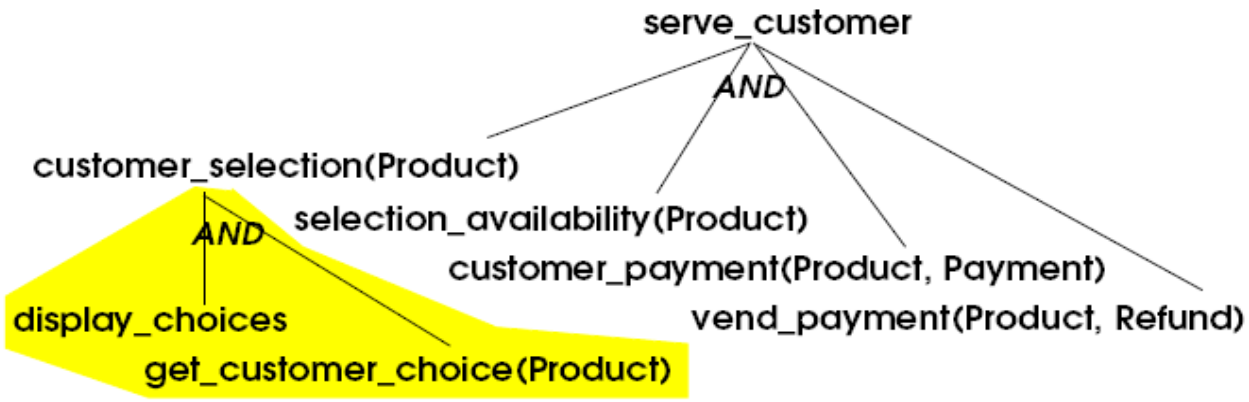
How to elicit?

Goal-Directed Strategy 1: Classical Logic Approach

- Example: "good old vending machine"

step 3: incremental expansion

E.g., the customer needs to know about the choices, and these choices should be displayed in some form; the machine should be able to accept the customer selection.



- ---> Executable specification (e.g., in Prolog)

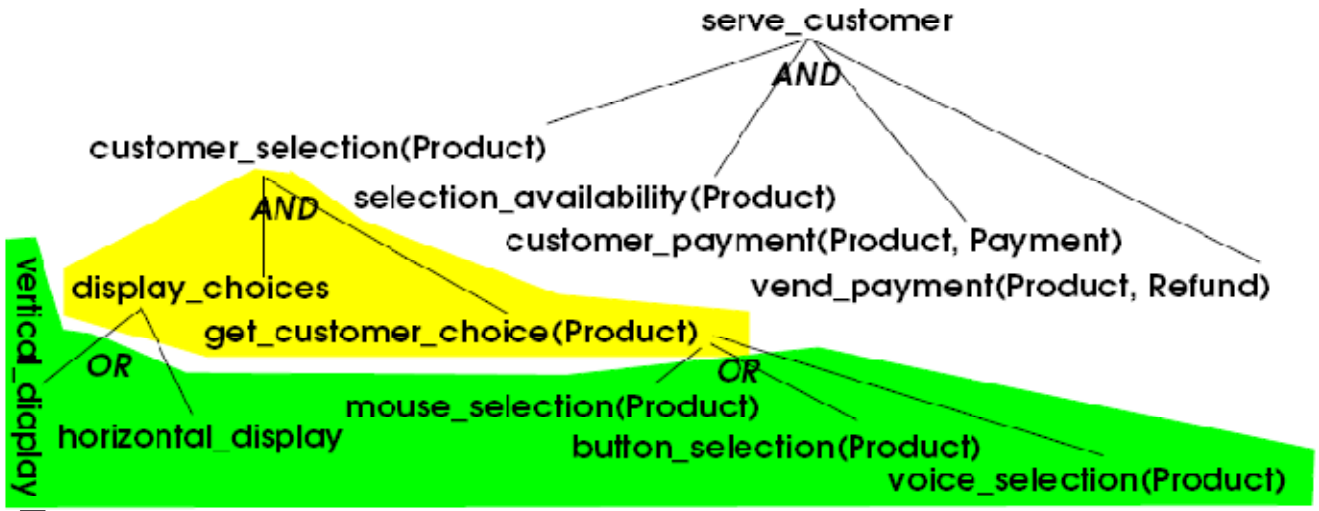
```
serve_customer :- customer_selection(Product),
                  selection_availability(Product),
                  customer_payment(Product, Payment),
                  vend_payment(Product, Refund).

customer_selection(Product) :- display_choices,
                               get_customer_choice(Product).
```

How to elicit?

Goal-Directed Strategy 1: Classical Logic Approach

F Example: "good old vending machine"

step 4: how-to elaboration



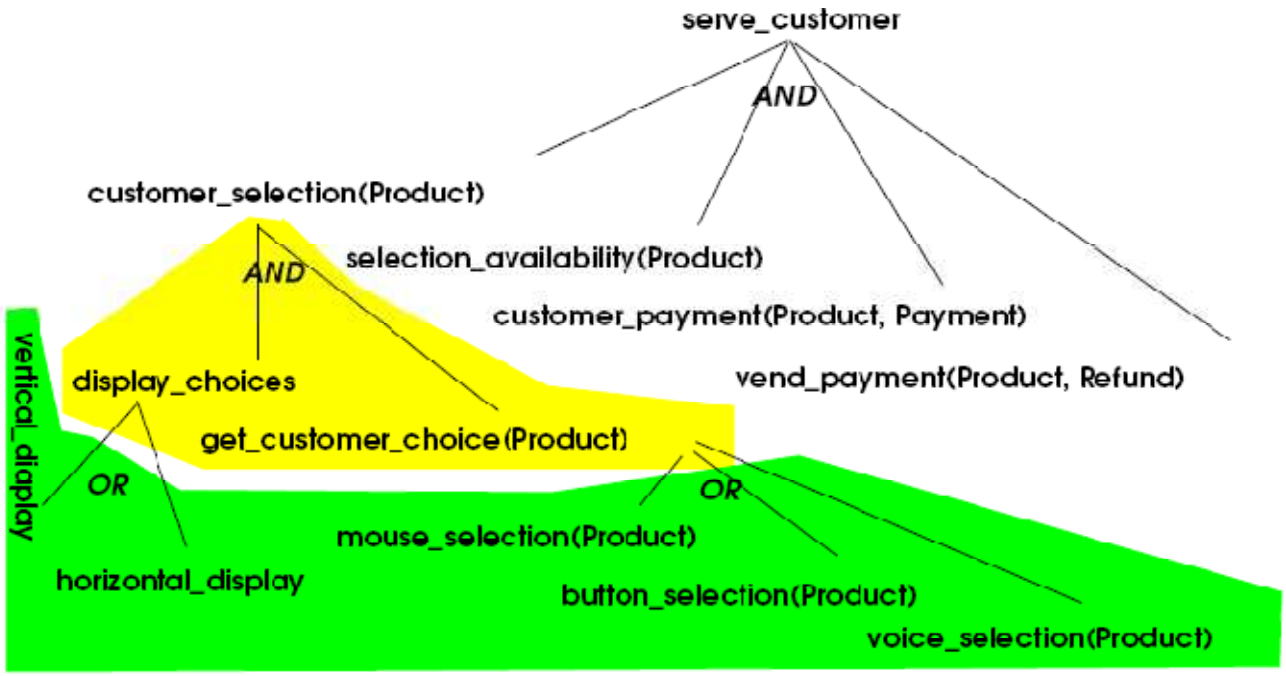
---> Executable specification (e.g., in Prolog)

```
serve_customer :- customer_selection(Product) ,
                  selection_availability(Product) ,
                  customer_payment(Product, Payment) ,
                  vend_payment(Product, Refund) .

customer_selection(Product) :- display_choices ,
                               get_customer_choice(Product) .

display_choices :- vertical_diaplay .
display_choices :- horizontal_display .
```

How to elicit?

Goal-Directed Strategy 1: Classical Logic Approach

F Example: "good old vending machine"



- => Dependency => Traceability:
  - From design to requirements (avoid any erroneous, accidental design)
  - From requirements to design(ensure every requirement is met)
- => Rational Design
  - basis for alternatives, tradeoffs, rationale

Lawrence Chung

More on Traceability

=> Dependency => Traceability:

From design to requirements (avoid any erroneous, accidental design)

From requirements to design(ensure every requirement is met)

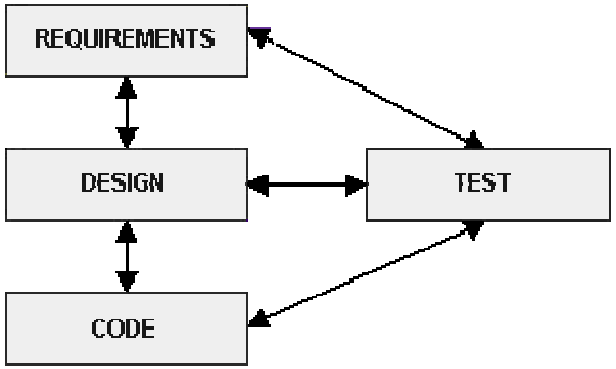
*What would be "**forward traceability**"?*

*What would be "**backward traceability**"?*

Traceability matrix vs. graph-oriented traceability?

Requirements Traceability Matrix

- A traceability matrix is created by associating requirements with the work products that satisfy them. Tests are associated with the requirements on which they are based and the product tested to meet the requirement.
  [http://www.jiludwig.com/Traceability_Matrix_Structure.html]



Sample Traceability Matrix

W, R, or S?

ID	USER REQUIREMENTS	FORWARD TRACEABILITY
U2	Users shall process retirement claims.	S10, S11, S12
U3	Users shall process survivor claims.	S13

ID	FUNCTIONAL REQUIREMENTS	BACKWARD TRACEABILITY
S10	The system shall accept requirement data.	U2
S11	The system shall calculate the amount of retirement.	U2
S12	The system shall calculate point-to-point travel time.	U2
S13	The system shall calculate the amount of survivor annuity	U3

Eliminate, rewrite, or correct traceability?

Requirements Traceability Matrix: Variations

http://departmentforms.dpsk12.org/dots/smedocs/requirements%20traceability%20matrix.pdf

Requirements Traceability Matrix

User Requirements (List the title or number of the user functional requirement)	**System Requirements** (When applicable, list the title or number title of the corresponding system requirement)	**Design Specification** (When applicable, list any specifications which must be satisfied by the design)	**Coding Component Reference** (When applicable, code units, subroutines, or modules which implement the requirement)	**Integration or System Test Case Reference** (Include number of the test script for the requirement)

http://www.fns.usda.gov/wic/StateInformationSystems/FReD/AppendixAFRED-ERTMVersion1.0Summer2002.pdf

Requirements Elicitation: Part II

- Why is it difficult?
  - Critical Issues
- What to elicit?
  - Four Worlds of RE
  - The Reference Model Revisited
- How to elicit?
  - Goal-Oriented Elicitation
    - Classical Logic Approach
    - Traceability
    - With A Richer Ontology
  - Domain Analysis
  - Problem Frames
  - Data/Information Elicitation Techniques

How to elicit?

Goal-Directed Strategy 2: Using more expressive power

Expressive Power *Revisited*

"Propositional and predicate logic provide all the basic concepts needed for a systematic engineering design methodology"

[C. A. Hoare, Mathematical Models for Computing Science, Oxford Univ. Rpt., Aug. 1994]

- **modelling all the possible worlds**

  => what are they?

  => what are our conceptualization of them?

  these are philosophical questions!

- **Ontology**

  on-tol-o-gy n. The branch of philosophy that deals with being

  what exists in reality?

  what are essential things in reality?

  entities, activities, constraints

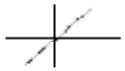
  goals, agents, roles, rationales

- **Epistemology**

  e-pis-te-mol-o-gy n., pl. -gies. 1. The division of philosophy that investigates the nature and origin of knowledge.

  2. A theory of the nature of knowledge.

  how do we organize them?



How to elicit?

- Goal-Directed Strategy 2: Using more expressive power
  - Example: A Library System

[Dardenne, van Lamsweerde & Fickas, "Goal-directed Requirements Acquisition," Science of Computer Programming, 20, pp. 3-50]

KAOS

- **Background**
  - Developed in the early 90's
  - first major teleological requirements modeling language
  - full tool support available
  - has been applied to a number of industrial case studies
- **Two parts:**
  - Semi-formal goal structuring model
  - Formal definitions for each entity in (linear) temporal logic
    - Liveness – Maintain: □(P→Q), Achieve: P→ ◊Q
    - Safety – Avoid: □(P→ ¬ Q)
- **Approach**
  - **Method focuses on goal elaboration:**
    - define initial set of high level goals & objects they refer to
    - define initial set of agents and actions they are capable of
  - **Then iteratively:**
    - refine goals using AND/OR decomposition
    - identify obstacles to goals, and goal conflicts
    - operationalize goals into constraints (or sw requirements) that can be assigned to individual agents
    - refine & formalize definitions of objects & actions
    - Goal refinement ends when every subgoal is realizable by some individual agent assigned to it, that is, expressible in terms of conditions that are *monitorable* and *controllable* by the agent.

Prior's Tense logic

Pp	**◊t(t<now ∧ p(t))**
Fp	**◊t(now<t ∧ p(t))**
Hp	**∀t(t<now → p(t))**
Gp	**∀t(now<t → p(t))**

□: necessary
e.g., □ Fp
◊: possible
e.g., ◊ Fp

Extensions to Tense Logic

Spq	q has been true since a time when p was true
Upq	q will be true until a time when p is true

more in the module on Model Checking

How to elicit?

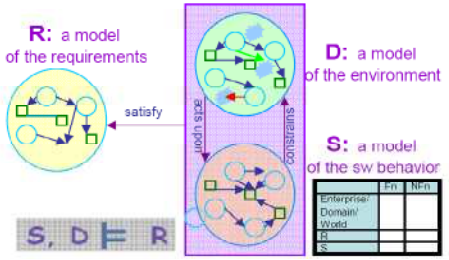
Goal-Directed Strategy 2: Using more expressive power

Example: A Library System

[Dardenne, van Lamsweerde & Fickas, "Goal-directed Requirements Acquisition," Science of Computer Programming, 20, pp. 3-50]

KAOS

- A ***goal*** is a prescriptive statement of intent about some system (existing or to-be) whose satisfaction in general requires the cooperation of some of the agents forming that system (= software and environment).
- ***Domain properties*** are descriptive statements about the environment .e.g., physical laws, organizational norms, etc.
- A ***requirement*** is a *realizable* goal under responsibility of *an agent in the software-to-be* (expressed in terms of *monitored* and *controlled* objects);
- An *expectation* is a *realizable* goal under responsibility of *an agent in the environment* (unlike requirements, expectations cannot be enforced by the software-to-be).
- An ***operation*** is an input-output relation over objects
- The state of the system (software or environment) is defined by aggregation of the states of its objects. An ***object model*** is represented by a UML class diagram.



If ***R*** denotes the set of requirements,
As the set of assumptions *(expectations)*,
S the set of software specifications,
Ac the set of accuracy goals, and
G the set of goals,
the following satisfaction relations must hold:

$S, Ac, D \models R$ with $S, Ac, D \not\models$ **false**

$R, As, D \models G$ with $R, As, D \not\models$ **false**

Although both optative, requirements are to be enforced by the software whereas assumptions/expectations can be enforced by agents in the environment only

How to elicit?

Goal-Directed Strategy 2: Using more expressive power

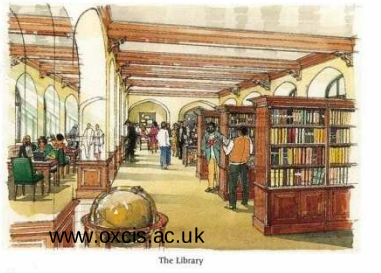
Example: A Library System

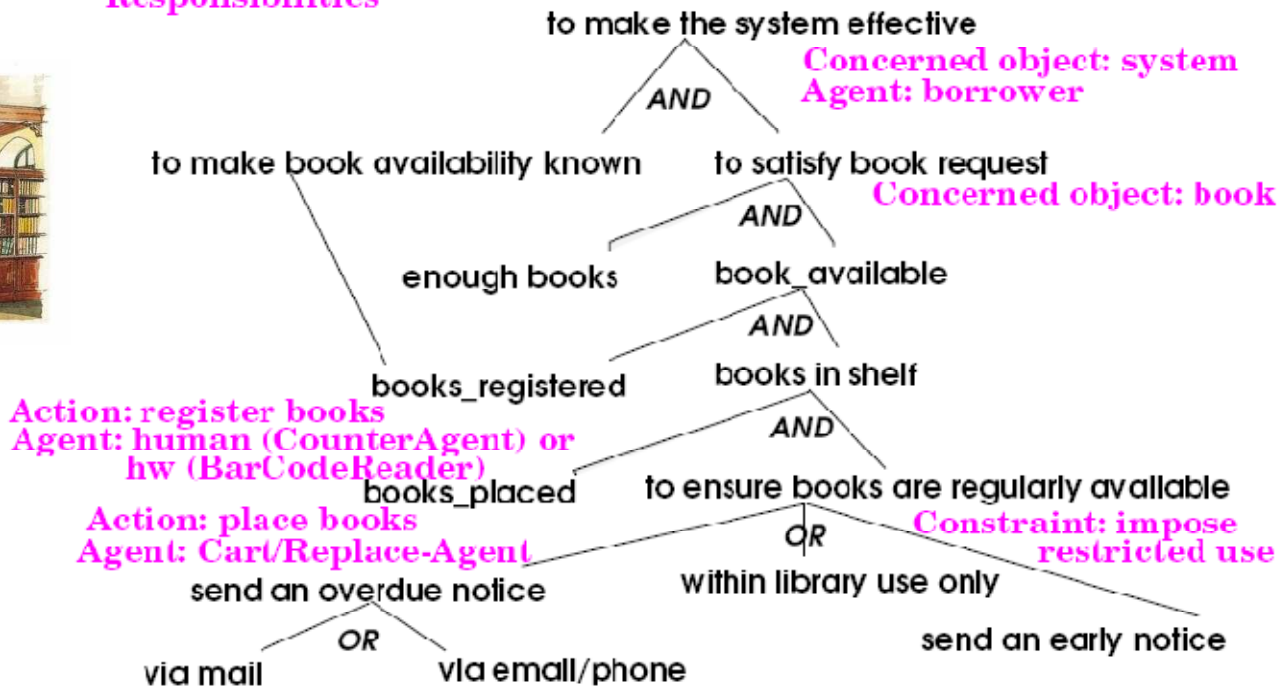
[Dardenne, van Lamsweerde & Fickas, "Goal-directed Requirements Acquisition, Science of Computer Programming, 20, pp. 3-50]

Richer ontology -> during goal reduction, identity:

- **Concerned objects**
- **Constraints**
- **Actions**
- **Agents (human, hw, sw,...)**
- **Responsibilities**

Leaf goals are "operationalized" in terms of Agents/Constraints

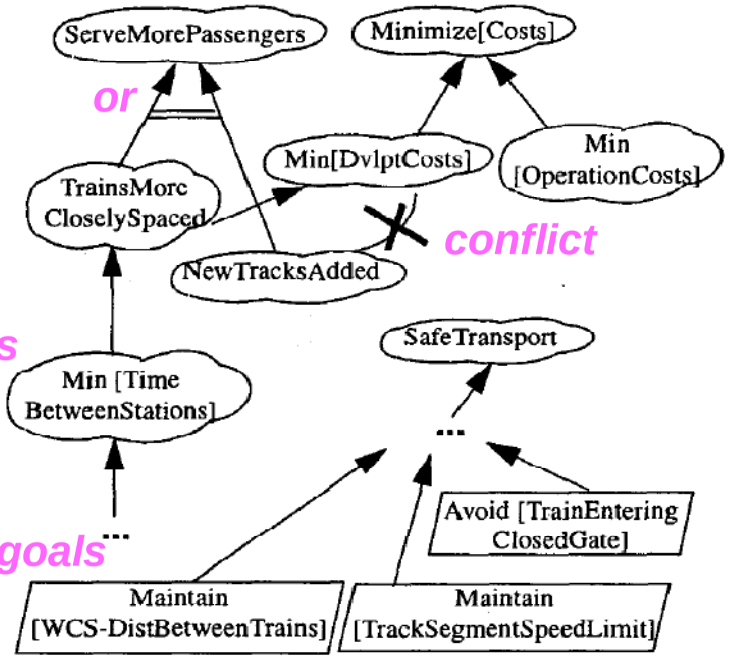




KAOS: Automated Train Control System Example

[A. van Lamsweerde, "Requirements engineering in the year 00: a research perspective", *Proc., 22nd ICSE'00*, pp5-19. IEEE Computer Society Press]

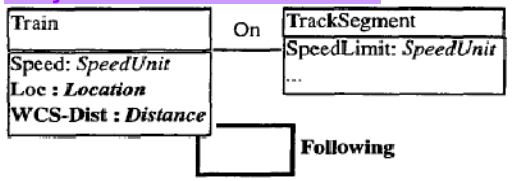
Maintain, Avoid: "always" goals $\Box (P \rightarrow Q)$ $\Box (P \rightarrow \neg Q)$.
Achieve: "eventually" $P \Rightarrow \Diamond Q$.
"=>": logical implication (the two below are the same)
$P \Rightarrow Q$ $\Box (P \rightarrow Q)$



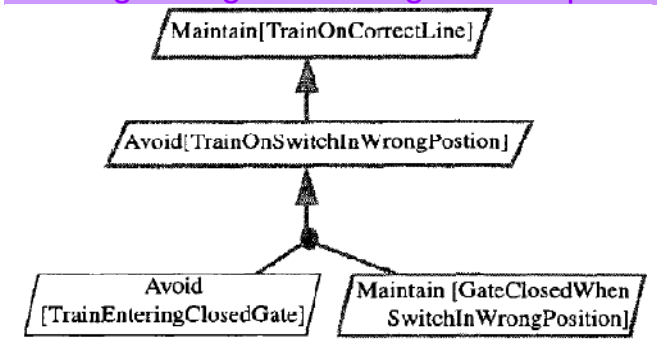
Goal Maintain[TrackSegmentSpeedLimit]
InformalDef *A train should stay below the maximum speed the track segment can handle.*
FormalDef ∀ tr: Train, s: TrackSegment :
On(tr, s) ⇒ tr.Speed ≤ s.SpeedLimit

Goal Maintain[WCS-DistBetweenTrains]
InformalDef *A train should never get so close to a train in front so that if the train in front stops suddenly (e.g., derailment) the next train would hit it.*
FormalDef ∀ tr1, tr2: Train :
Following(tr1, tr2) ⇒ tr1.Loc - tr2.Loc > tr1.WCS-Dist
worst case stopping

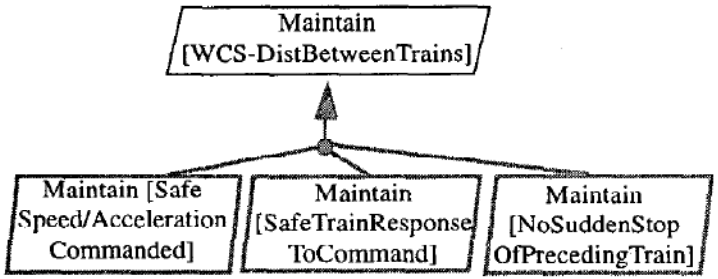
Object model



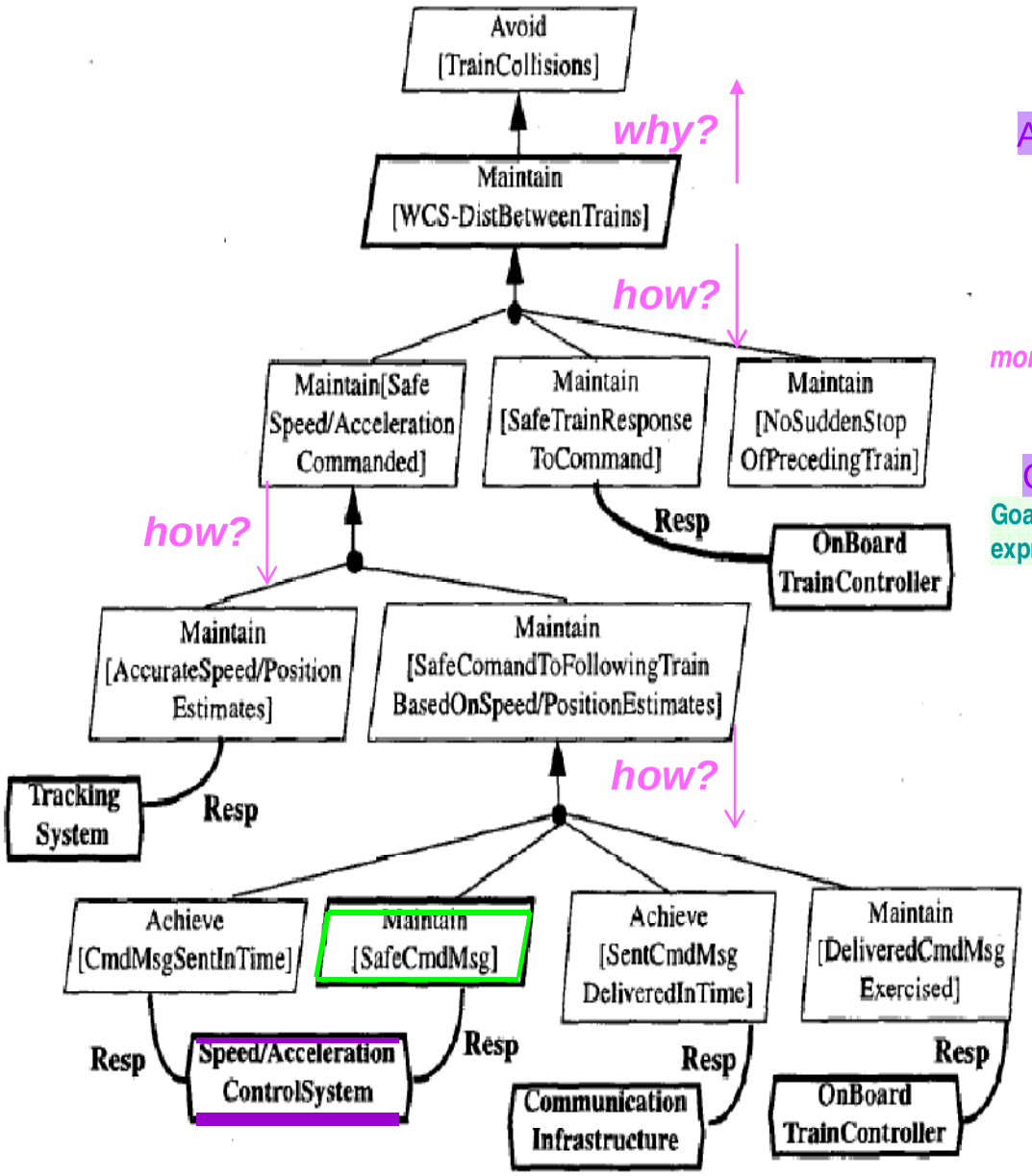
Eliciting new goals through WHY questions



Eliciting new goals through HOW questions



KAOS: Automated Train Control System Example



Goal Maintain[SafeCmdMsg] — *monitored*

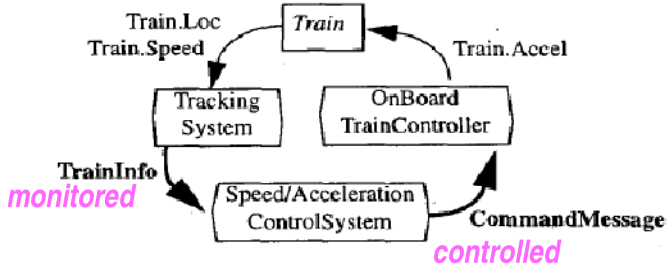
FormalDef ∀ cm: *CommandMessage*, ti1, ti2: *TrainInfo*

cm.Sent ∧ cm.TrainID = ti1.TrainID ∧ FollowingInfo (ti1, ti2)

⇒ cm.Accel ≤ F (ti1, ti2) ∧ cm.Speed > G (ti1)

controlled

Agent interfaces



Operations, Operationalizations

Goals refer to specific state transitions; Operations causing them are expressed by domain pre- and postconditions. For Maintain[SafeCmdMsg]:

Operation SendCommandMessage
Input Train {**arg** tr}
Output ComandMessage {**res** cm}
DomPre ¬ cm.Sent
DomPost cm.Sent ∧ cm.TrainID = tr.ID

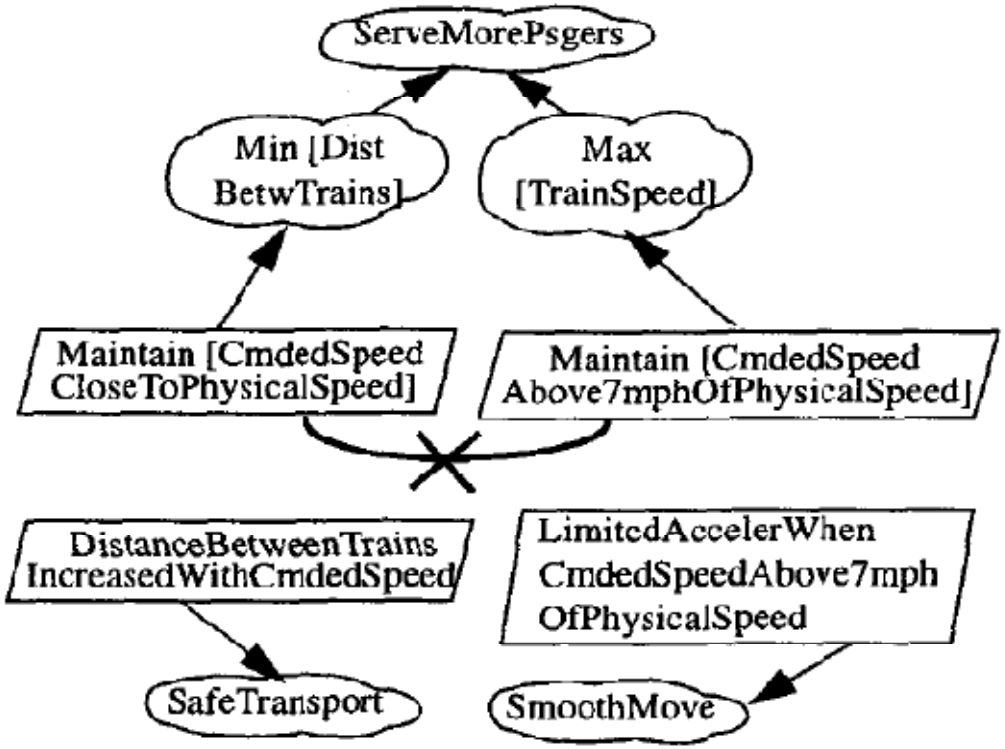
Strengthen domain conditions so that the various goals linked to the operation are ensured. For goals assigned to software agents, this step produces requirements on the operations for the corresponding goals to be achieved.

Operation SendCommandMessage
Input Train {**arg** tr}, TrainInfo; **Output** ComandMsg {**res** cm}
DomPre ... ; **DomPost** ...
ReqPost for SafeCmdMsg:
Tracking (ti1, tr) ∧ Following (ti1, ti2)
→ cm.Acc ≤ F (ti1, ti2) ∧ cm.Speed > G (ti1)
ReqTrig for CmdMsgSentInTime:
■≤0.5 sec ¬ ∃ cm2: CommandMessage:
cm2.Sent ∧ cm2.TrainID = tr.ID

g

KAOS: Automated Train Control System Example

[A. van Lamsweerde, "Requirements engineering in the year 00: a research perspective", *Proc., 22nd ICSE'00*, pp5-19. IEEE Computer Society Press]



Conflicts

Conflicting Goals:

Goal Maintain [CmdedSpeedCloseToPhysicalSpeed]
FormalDef $\forall$ tr: Train
$tr.Acc_{CM} \geq 0$
$\Rightarrow tr.Speed_{CM} \leq tr.Speed + f(dist\text{-}to\text{-}obstacle)$

Goal Maintain [CmdedSpeedAbove7mphOfPhysicalSpeed]
FormalDef $\forall$ tr: Train
$tr.Acc_{CM} \geq 0 \Rightarrow tr.Speed_{CM} > tr.Speed + 7$

Boundary condition for logical inconsistency

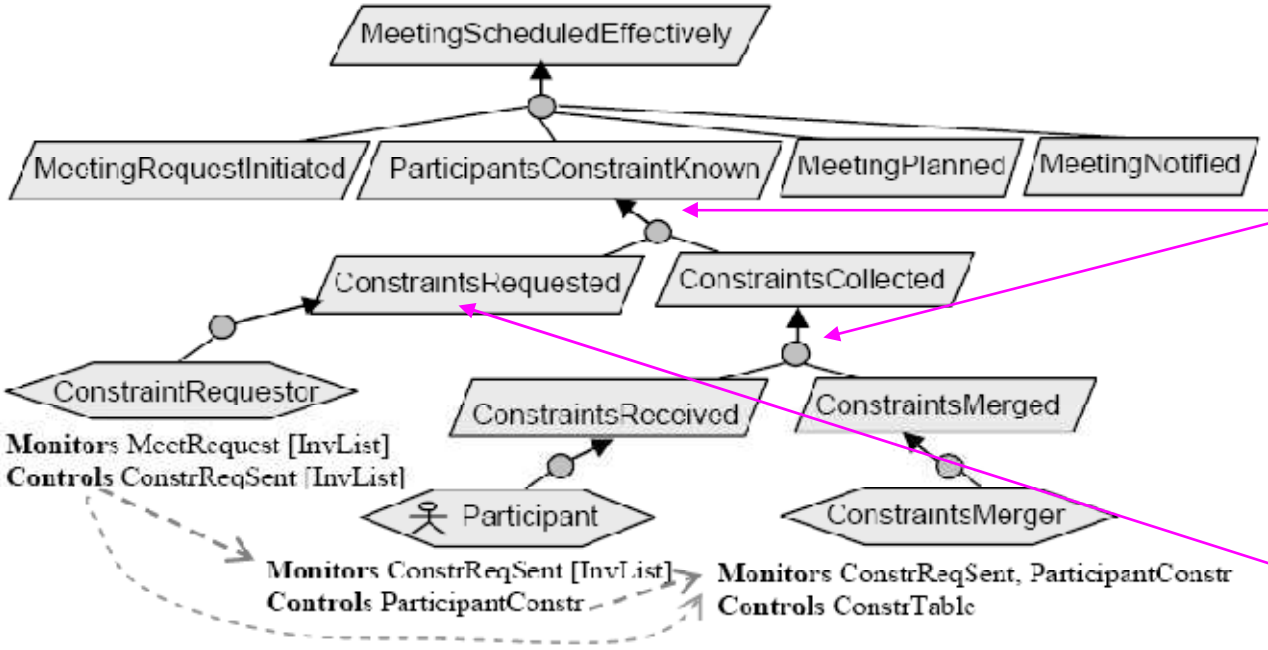
$\Diamond (\exists$ tr: Train$) (tr.Acc_{CM} \geq 0 \wedge f(dist\text{-}to\text{-}obstacle) \leq 7)$

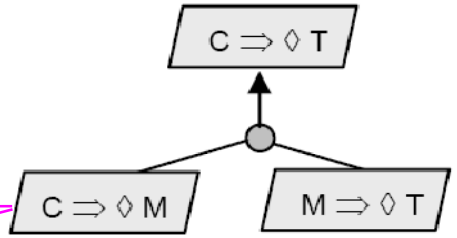
Conflict resolution:
e.g., keep the safety goal and weaken the other

Goal Maintain [CmdedSpeedAbove7mphOfPhysicalSpeed]
FormalDef $\forall$ tr: Train
$tr.Acc_{CM} \geq 0 \Rightarrow tr.Speed_{CM} > tr.Speed + 7$
$\vee\ f(dist\text{-}to\text{-}obstacle) \leq 7$

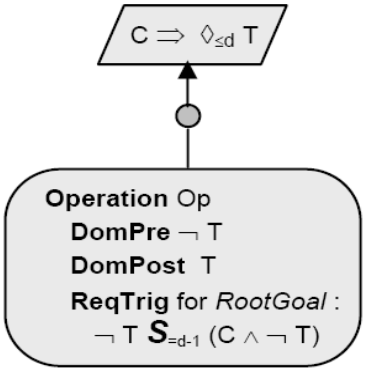
KAOS: Meeting Scheduler Example

[A. van Lamsweerde, From System Goals to Software Architecture", M. Bernardo and P. Inverardi (Eds.): SFM 2003, *LNCS 2804*, pp. 25-43, 2003.





Refinement-by-milestone pattern



S = Since

Bounded-Achieve operationalization pattern

Goal Achieve [ParticipantsConstraintsKnown]

FormalSpec ∀m: Meeting, p: Participant

Requested (m) ∧ Invited (p, m) ∧ Scheduling (s, m) ⇒ ◊≤Md Knows (s, p.Constraints)

Requirement Achieve [ConstraintsRequested]

FormalSpec ∀m: Meeting, p: Participant:

Requested (m) ∧ Invited (p, m) ⇒ ◊≤Rd ConstrRequested (p)

The date constraints of people expected to attend a meeting shall be known to the scheduler within M days after the meeting is requested:

Specification:

Any issues?

∀m: MeetingClass, p: ParticipantClass

MeetRequest (m) ∧ p **in** InviteeList (m) ⇒ ◊≤Rd ConstrReqSent (p)

Accuracy Goal:

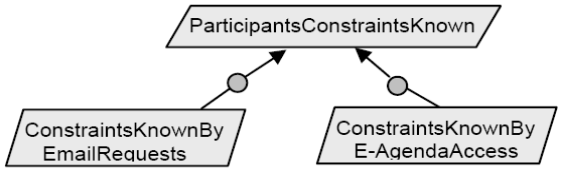
∀m: Meeting, m': MeetingClass, p: Participant, p': ParticipantClass

Mapping (m, m') ∧ Mapping (p, p') ⇒

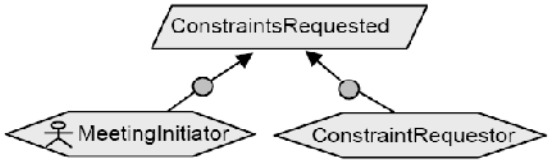
MeetRequest (m) ⇔ Requested (m)

p' in InviteeList (m') ⇔ Invited (p, m)

ConstrReqSent (p) ⇔ ConstrRequested (p)



Alternative goal refinements



Alternative agent assignments

The NFR Framework

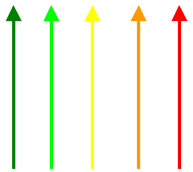
J. Mylopoulos, L. Chung, E. Yu, "From object-oriented to goal-oriented requirements analysis ", *CACM*, pp31-37.ACM Press

Goal-oriented analysis focuses on the description and evaluation of alternatives and their relationship to the organizational objectives.

Non-functional Requirements: softgoal interdependency graph (SIG) ➔ See a separate Module & [CNYM2000]

softgoal – to be satisficed, i.e., achieved but not in the absolute sense; uncertainty in problem, uncertainty in solution ***(uncertainty in designation***

Contributions:



AND $(G,\{G_1,G_2,\ldots,G_n\})$	-- goal G is satisficed when all of $G_1,G_2,\ldots,G_n$ are satisficed and there is no negative evidence against it; -- goal G is unsatisficed and there is one of $G_1,G_2,\ldots,G_n$ is unsatisficed and there is no positive evidence for it.
OR $(G,\{G_1,G_2,\ldots,G_n\})$	-- goal G is satisficed when one of $G_1,G_2,\ldots,G_n$ is satisficed and there is no negative evidence against it; -- goal G is unsatisficeable if all of $G_1,G_2,\ldots,G_n$ are unsatisficeable and there is no positive evidence for it.
$+(G_1,G_2)$	-- goal G_1 contributes positively to the satisficing of goal G_2.
$-(G_1,G_2)$	-- goal G_1 contributes negatively to the satisficing of goal G_2.

From NFR Softgoals to Operationalizing Softgoals (e.g., Use Cases)

operationalization – softgoals are achieved thru (actor-)operations (e.g., in terms of use cases)

Object model

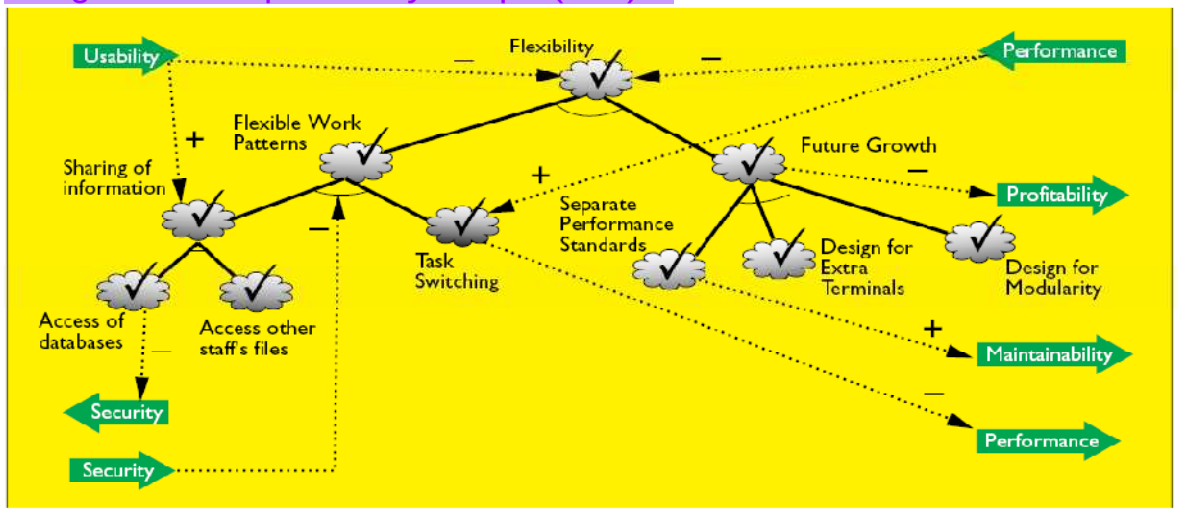
entities (and their relationships) and activities (and their relationships) in the domain are represented in terms of objects, object structures, interactions and behavior

The NFR Framework

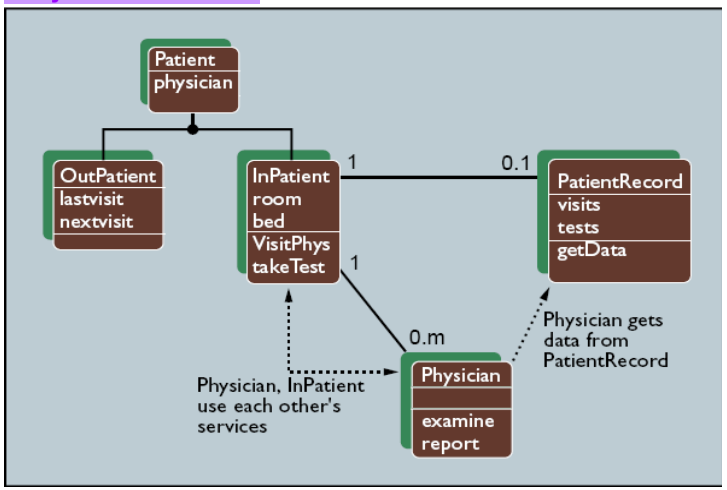
J. Mylopoulos, L. Chung, E. Yu, "From object-oriented to goal-oriented requirements analysis ", *CACM*, pp31-37.ACM Press

Example: A small portion of a hospital model for requirements analysis

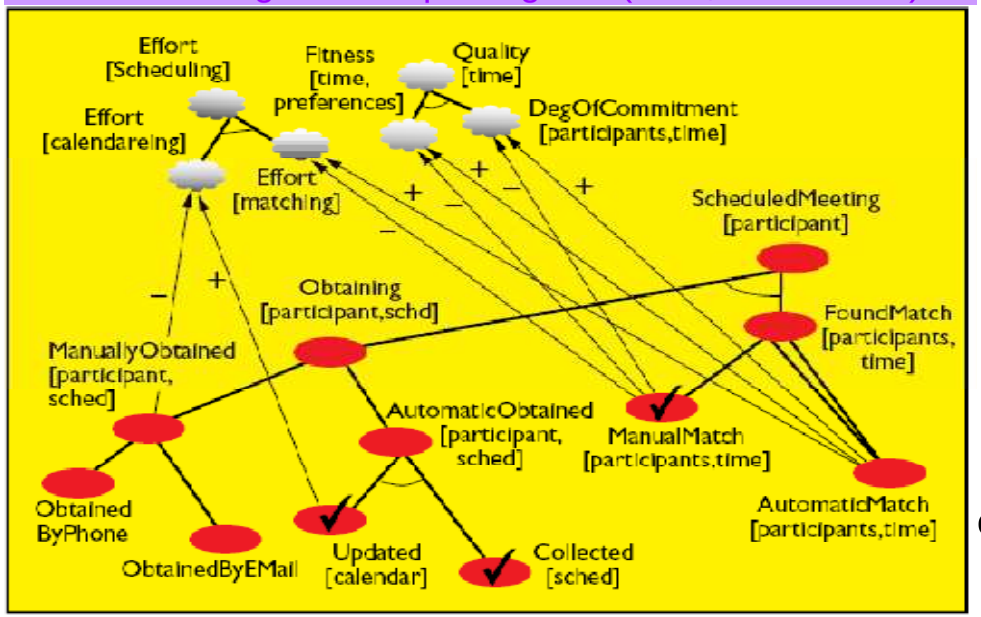
Softgoal Interdependency Graph (SIG)



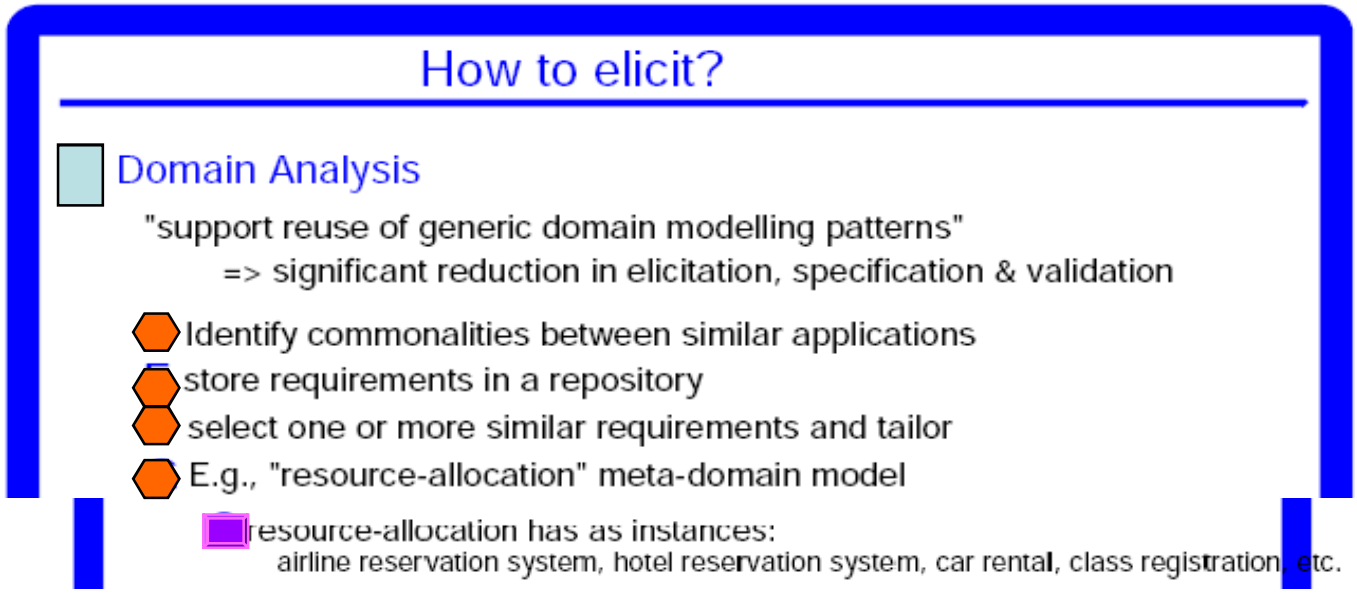
Object Model



From NFR Softgoals to Op Softgoals (here, Use Cases)



e Chung

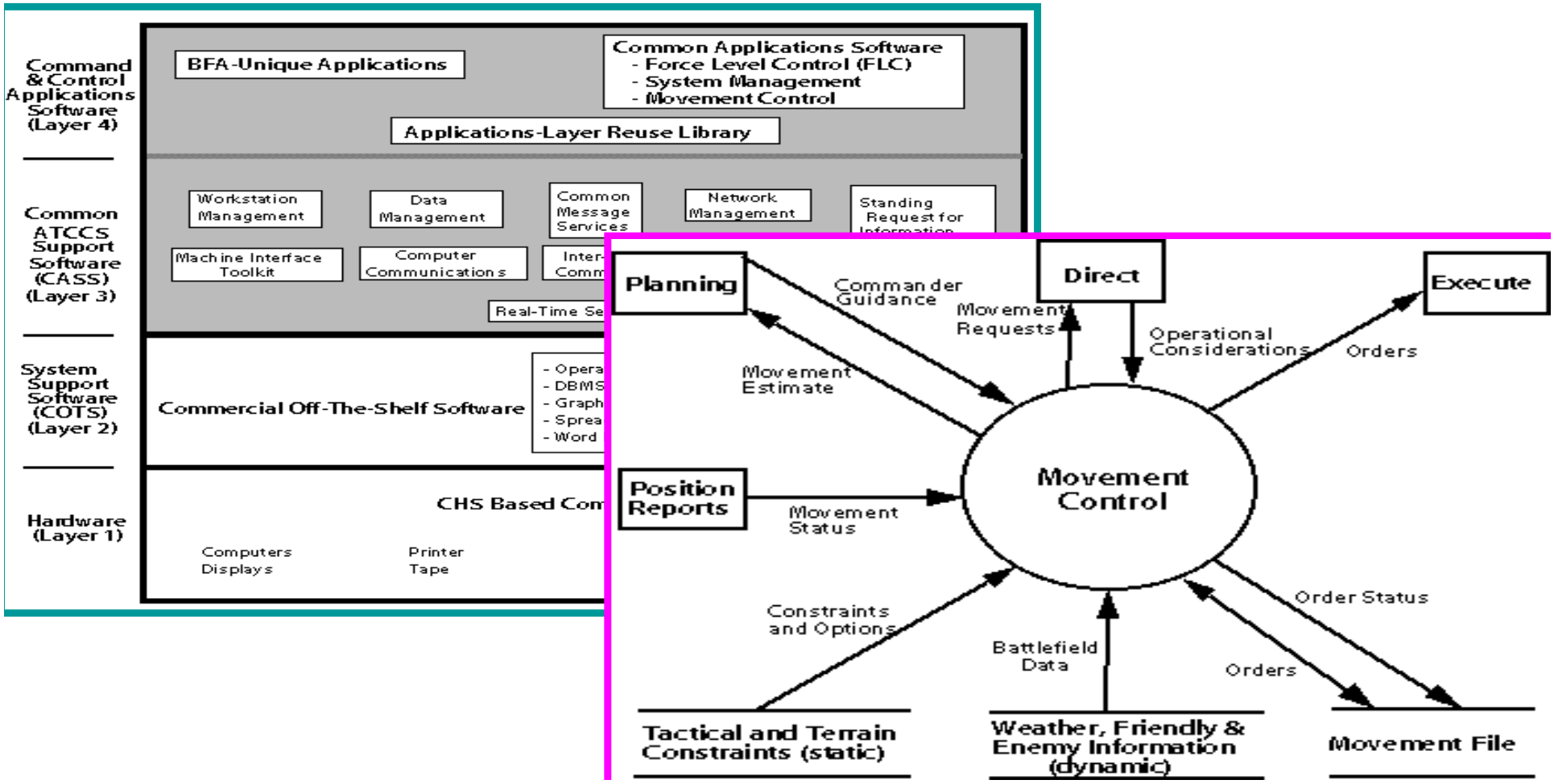


- Feature Oriented Domain Analysis (FODA): a process for domain analysis and establishes specific product for later use. Three basic phases:
  - **Context Analysis:** defining the extent (or bounds) of a domain for analysis
  - **Domain Modeling:** providing a description of the problem space in the domain that is addressed by software (See ***Enterprise Modeling***)
  - **Architecture Modeling:** creating the software architecture(s) that implement solutions to the problems in the domain
    - Note: The architectural modeling phase was initially defined as part of the FODA methodology. However, the process of integrating FODA products with architectural modeling has become part of the domain design activity in the overall concept of Domain Engineering.

[CMU/SEI-90-TR-21].

How to Elicit?: FODA

- **Context Analysis:** [adapted from CMU/SEI-90-TR-21].
  - **Structure Diagram -** an informal block diagram in which the domain is placed relative to higher-, lower-, and peer-level domains.
  - **Context Diagram -** a data flow diagram showing data flows between a generalized application within the domain and the other entities and abstractions with which it communicates.



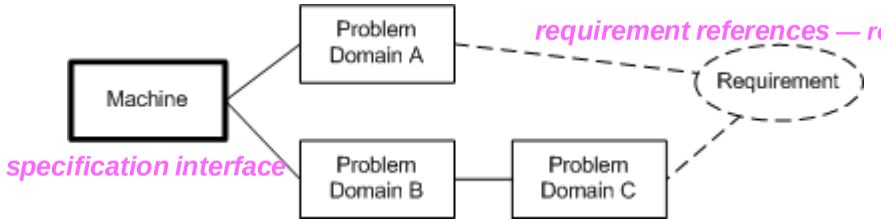
Problem Frames

[M. Jackson, *Problem Frames: Analysing and Structuring Software Development Problems,* 2001]

- *A problem analysis* approach in gathering requirements and creating specifications.
- *Fundamental Philosophy:*
  - Requirements analysis should be through a process of parallel, not hierarchical, decomposition of user requirements.
  - User requirements are about relationships in the operational context; not about functions that the software system must perform.

Problem Diagrams

requirements analyst: develop a specification for the behavior that the Machine must exhibit at the Machine interface in order to satisfy the requirement.

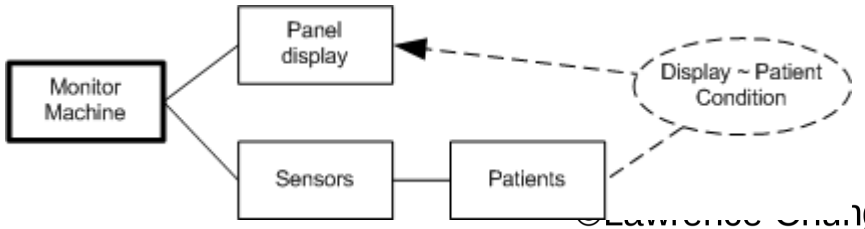


requirement references — references in the requirement to phenomena in the problem domains

specification interface

Example:

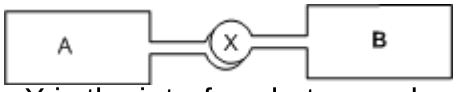
Requirement: The panel display must display information that matches and accurately reports the condition of the patients.



Problem Frames

- *A problem frame is a description of a recognizable class of problems, where the class of problems has a known solution. In a sense, problem frames are problem patterns.*

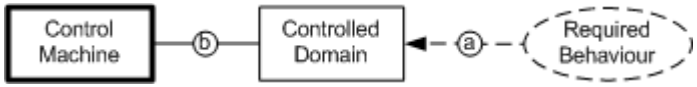
Notation: domain interface



X is the interface between domains A and B. Individuals that exist or events that occur in X, exist or occur in both A and B.

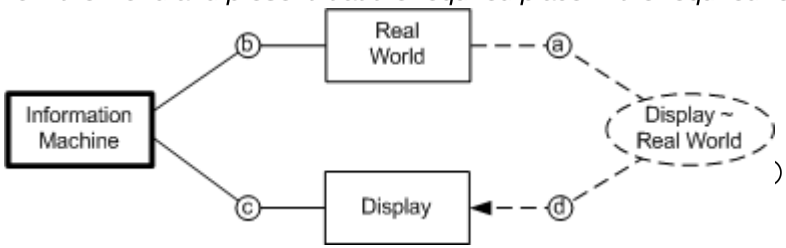
Required Behavior Problem Frame

There is some part of the physical world whose behavior is to be controlled so that it satisfies certain conditions.
The problem is to build a machine that will impose that control.



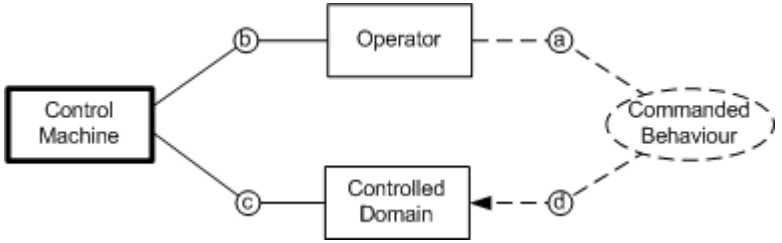
Information Display Problem Frame

There is some part of the physical world about whose states and behavior certain information is continually needed.
The problem is to build a machine that will obtain this information from the world and present it at the required place in the required form.



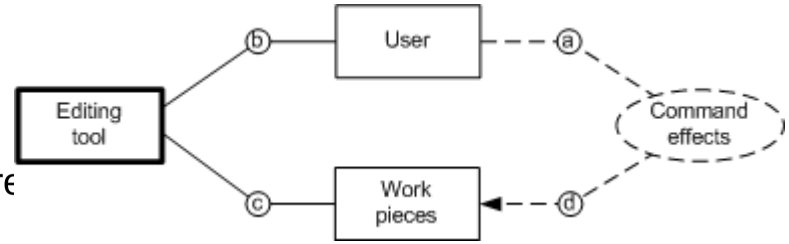
Commanded Behavior Problem Frame

There is some part of the physical world whose behavior is to be controlled in accordance with commands issued by an operator.
The problem is to build a machine that will accept the operator's commands and impose the control accordingly.



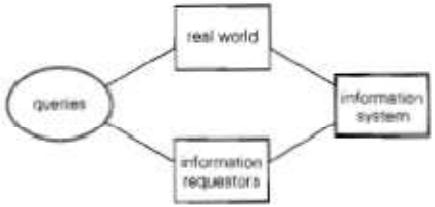
Simple Workpieces Problem Frame

A tool is needed to allow a user to create and edit a certain class of computer-processible text or graphic objects, or similar structures, so that they can be subsequently copied, printed, analyzed or used in other ways.
The problem is to build a machine that can act as this tool.

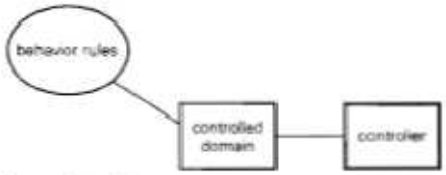


Problem Frames

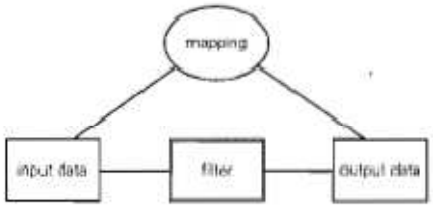
[B. L. Kovitz, "Practical Software Requirements: A Manual of Content & Style", pp.74-75]



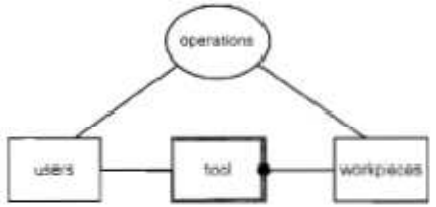
Information problem



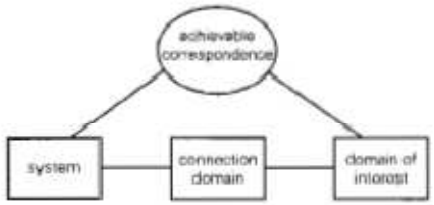
Control problem



Transformation problem



Workpiece problem



Connection problem

Anti-requirements, abuse frame, security requirements

Introducing Abuse Frames for Analysing Security Requirements,
Luncheng Lin, Bashar Nuseibeh, Darrel Ince, Michael Jackson, Jonathan Moffett
RE 2003.

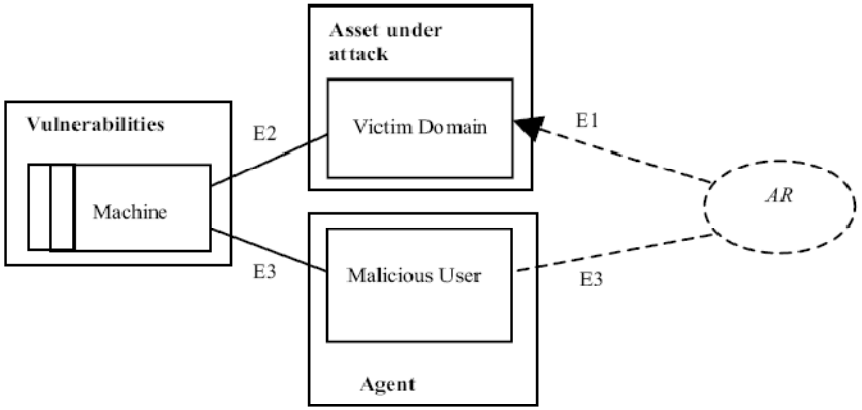


Figure 1: A threat described by a generic abuse frame diagram.

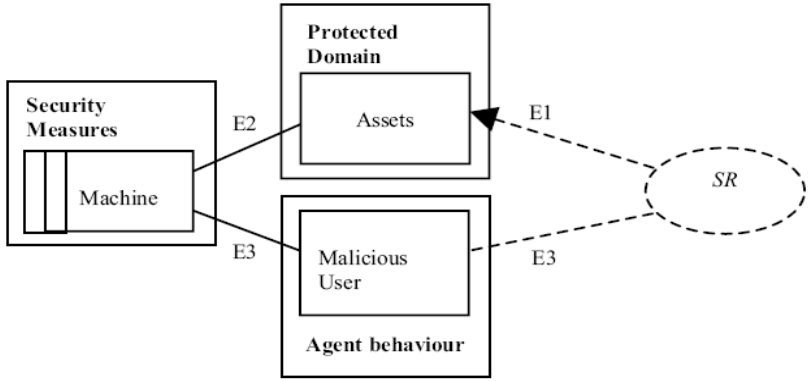


Figure 2: A security requirement expressed in Problem Frames.

Problem Frames

Source: Lecture note by Jim Herbsleb, http://conway.isri.cmu.edu/%7ejdh/MethodsF06/lec/probframes/prob-fr-1.pdf

The key idea in problem frames is *recurring problem types*

- Different problems share characteristics.
- If you stand back from the details of your current problem, you may recognize it as a known problem.
- Many known types of problems are already solved.
- A *problem frame* represents a "well-known" type of software problem.

Problem frames are a way of representing certain software-related expertise.

- *The nature of expertise*
  - Experts have about 50,000 chunks of knowledge
  - It takes them about 10 years to become experts
  - This is true across many domains
  - Experts recognize these chunks instead of deriving them
  - What passes for insight or intuition is often recognition
- *The idea in problem frames*
  - Frames represent chunks of knowledge
  - Knowing more variations will allow you to recognize more pre-solved problems and apply prior art
  - This is easier and less risky than analyzing from scratch

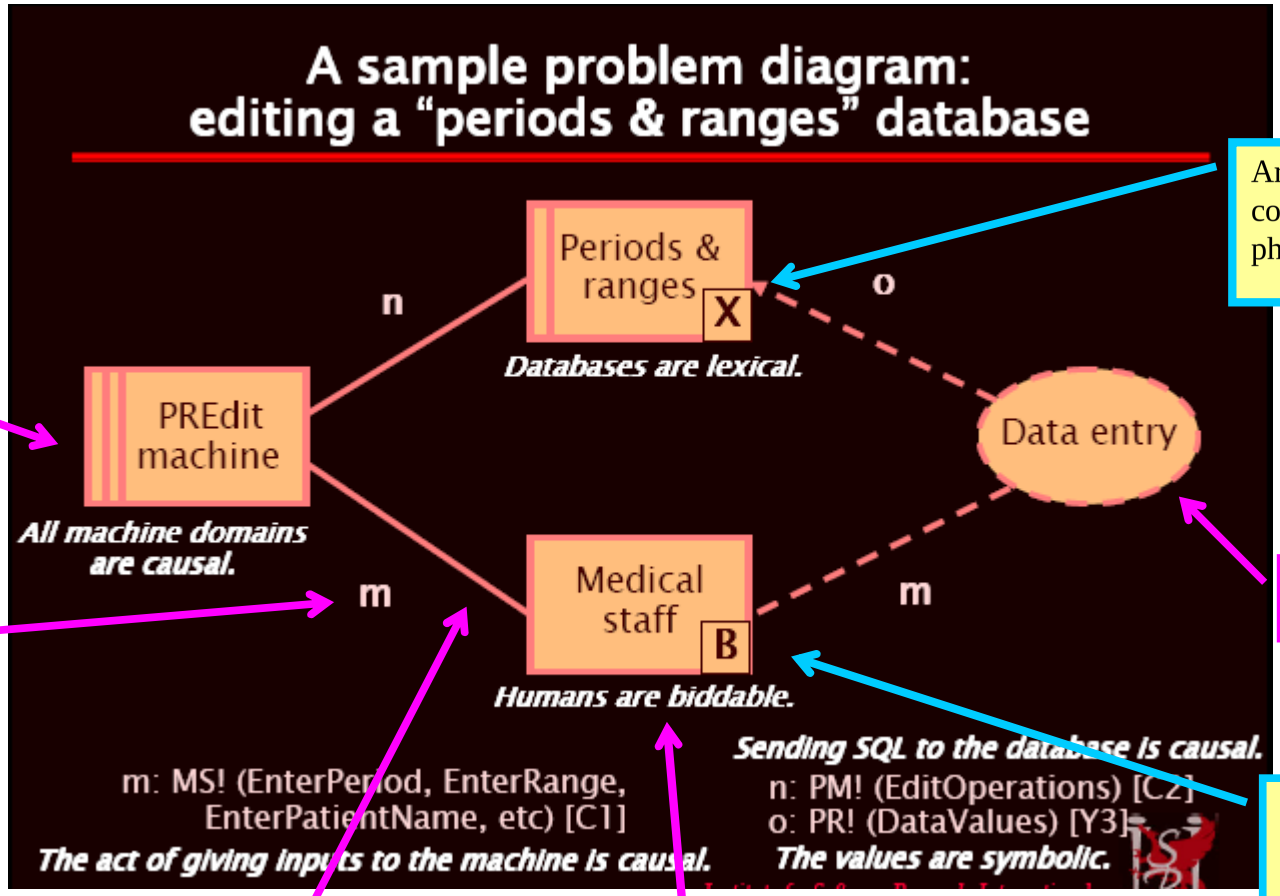
Problem Frames

Basic steps in applying problem frames

1. Break the context into pieces (called *domains*).
2. Identify the shared phenomena (called *interfaces*) among the domains.
3. Represent the domains and their interfaces in a *context diagram*
4. Add the conditions (called *requirements*) that the software must bring about.

- A context diagram that has been augmented with requirements is called a *problem diagram*.
- A problem diagram that recurs a lot is called a *problem frame*.

Problem Frames



Arrowed dashed line = constraining domain's phenomena

system

Shared phenomena

requirements

Dashed line = referencing domain's phenomena

interface

domain

Problem Frames

The different types of domains

- Causal (C) – has predictable relationships among physical phenomena
  - The machine domain, which has a double stripe, is always a causal domain.
- Biddable (B) - physical but unpredictable
  - Humans are the most common biddable domain.
- Lexical (X) – physical representation of data and symbolic phenomena
  - Designed domains, which have a single stripe, are usually lexical domains.

The different types of phenomena

- Causal (C or E) – the events that one domain initiates in order to influence or control another domain
  - Causal phenomena deal with what *happens*.
- Symbolic (Y) – values, truths, and states
  - Symbolic phenomena are *encodings* of other phenomena.

Examples of each type of phenomenon

- Causal (C or E)
  - DrawLine(x1,y1,x2,y2), supported by LCD screens
  - MakeElectricalPulse(voltage), supported by pacemakers
  - FireThruster(duration), supported by nuclear missiles
  - InsertRow(), supported by Microsoft Excel
  - ClickHyperlink(url), supported by humans
- Symbolic (Y)
  - ListOfParagraphs(), supported by Word documents
  - GridOfCells(), supported by spreadsheets
  - LevelOfEncryption(), supported by bank database

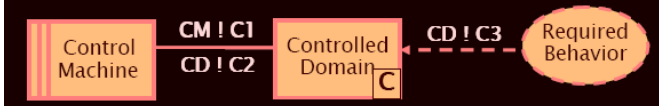
Problem Frames

There are 5 basic problem frames.

- Required behavior
  - Control part of the physical world to satisfy a condition.
- Commanded behavior
  - Control part of the physical world according to operator instructions.
- Information display
  - Obtain state/behavior information from the physical world and present it as required.
- Simple workpieces
  - Build a tool to create & edit persistent information objects
- Transformation
  - Transform information inputs to required outputs

Required behavior

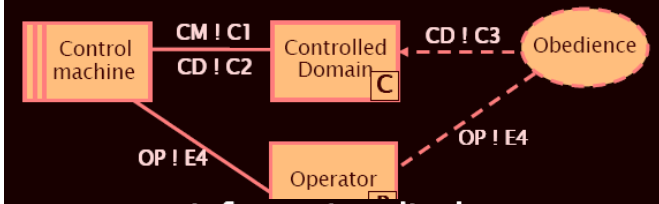
- Machine sends commands (C1) to controlled domain, which may provide feedback (C2)



- The requirement is that the controlled domain must demonstrate some behavior (C3).
- Note that the required behavior (C3) is different from the commands (C1) that are sent to try and cause that behavior.

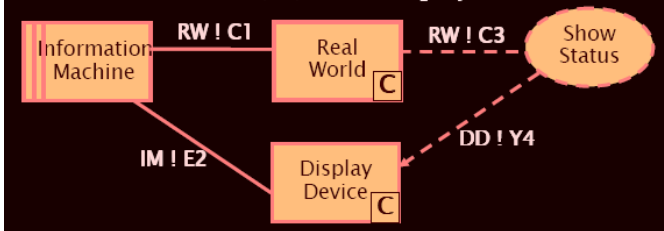
Commanded behavior

- Now a human operator sends certain events (E4) that implicitly specify the required behavior (C3).



Information display

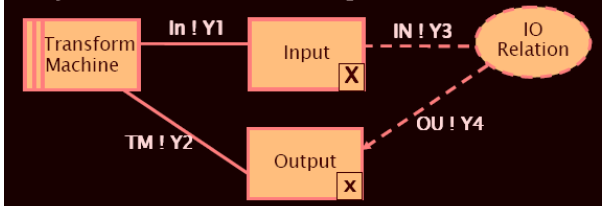
- The real world (or sometimes a lexical domain) sends some phenomena (C1) to the machine, which sends commands (E2) to the display device.



- Note that the real state of the world (C3) might not be entirely visible to the machine, yet the symbolic state (Y4) of the display should still be right!

Transformation

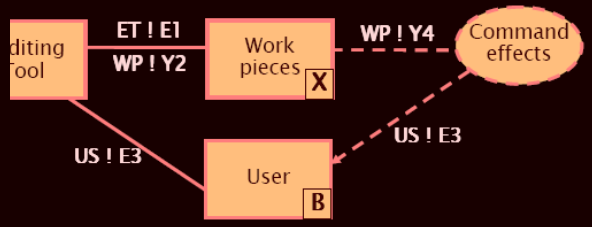
- The machine reads the input document's symbolic state (Y1) and actually constructs a totally new symbolic state (Y2) in an output document.



- Usually the symbolic states seen by the machine (Y1 and Y2) match the true states (Y3 and Y4).

Workpieces

- user's editing commands (E3) implicitly specify hat the document's symbolic state (Y4) should be.



- many cases, the symbolic state seen by the achine (Y2) matches the real symbolic state (Y4).

News article, 20 Oct 1992

AMBULANCE CHIEF QUITS AFTER PATIENTS DIE IN COMPUTER CRASH

By Ian MacKinnon and Stephen Goodwin

The Chief executive of the London Ambulance Service resigned yesterday over allegations that up to 20 people may have died because of the collapse of a new computer system controlling emergency calls. Virginia Bottomley, Secretary of Sate for Health, was forced to announce an external inquiry into the 36 hours over Monday and Tuesday which led to delays of up to three hours in ambulances arriving.

…

Source: Lecture note by Jim Herbsleb, http://conway.isri.cmu.edu/%7ejdh/MethodsF06/lec/probframes/prob-fr-3b.ppt

London Ambulance Manual System Problems

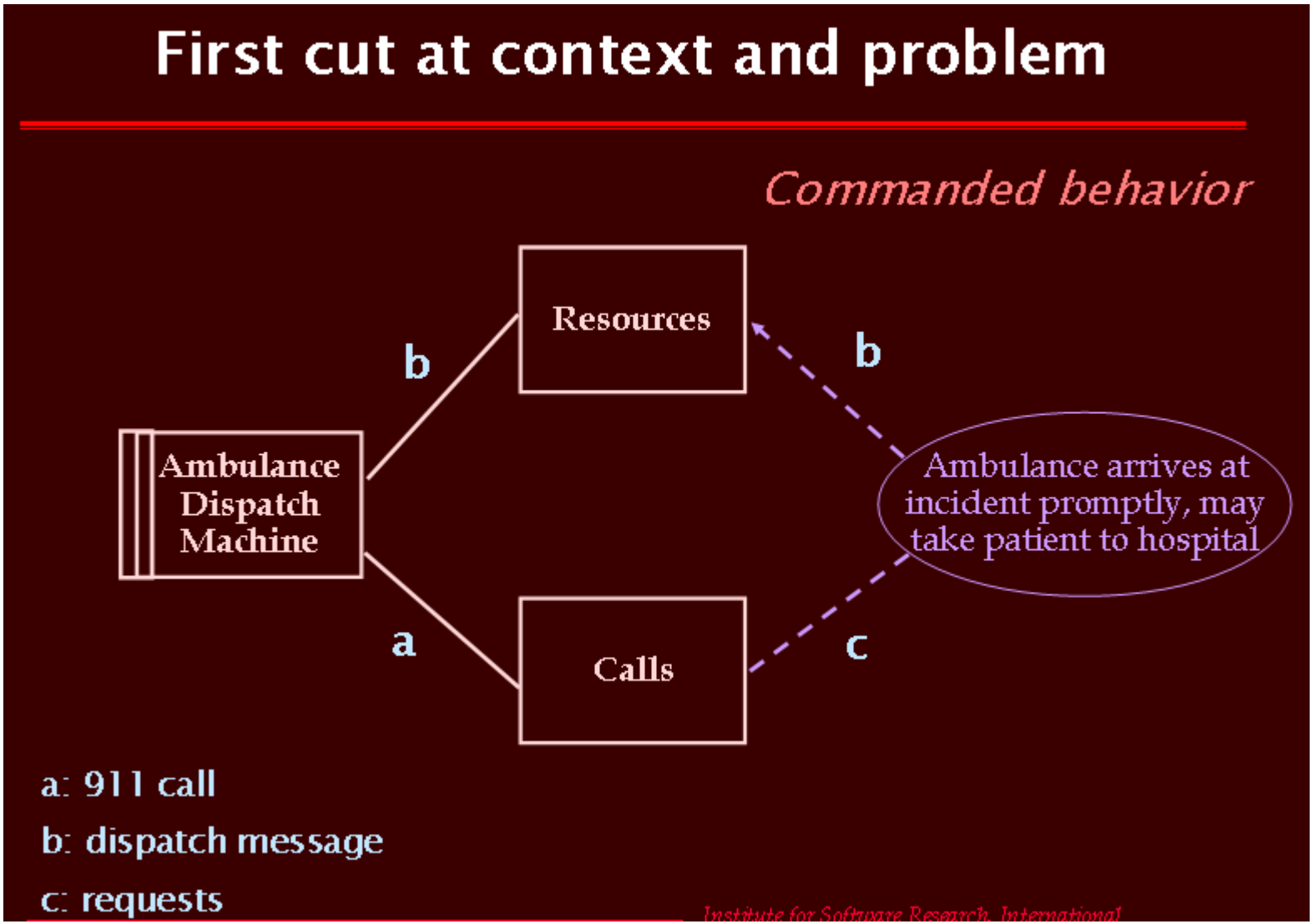
- identification of the precise location can be time consuming due to often incomplete or inaccurate details from the caller and the consequent need to explore a number of alternatives through the map books;
- the physical movement of paper forms around the Control Room is inefficient;
- maintaining up to date vehicle status and location information from allocators' intuition and reports from ambulances as relayed to and through the radio operators is a slow and laborious process;
- communicating with ambulances via voice is time consuming and, at peak times, can lead to mobilization queues;
- identifying duplicated calls relies on human judgment and memory. This is error prone;
- dealing with call backs is a labor intensive process as it often involves CA's leaving their posts to talk to the allocators;
- identification of special incidents needing a Rapid Response Unit or the helicopter (or a major incident team) relies totally on human judgment.

Source: Lecture note by Jim Herbsleb, http://conway.isri.cmu.edu/%7ejdh/MethodsF06/lec/probframes/prob-fr-3b.ppt

Critical Requirements

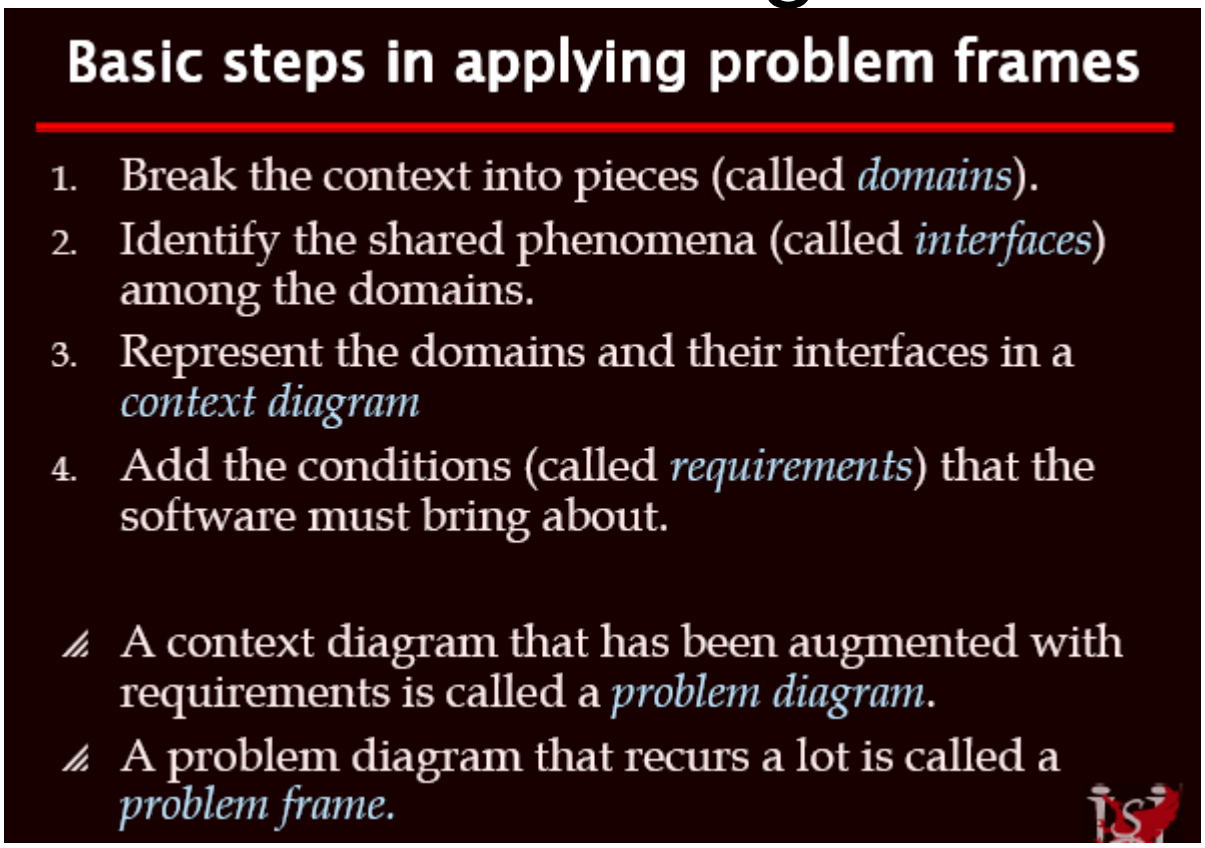
- Ambulance dispatch functionality
  - Calls report incidents and other needs for transport
  - An ambulance arrives at the location of an incident promptly; the ambulance may take patient(s) to hospital
- Other requirements
  - Timely response without communication overload
  - Resilience to faulty communication
  - Resilience to independent field decisions by personnel
  - Incremental information about incident
  - Efficient use of resources, efficient response
- System considerations
  - Incremental deployment
  - Fit with existing system processes

Source: Lecture note by Jim Herbsleb, http://conway.isri.cmu.edu/%7ejdh/MethodsF06/lec/probframes/prob-fr-3b.ppt



Source: Lecture note by Jim Herbsleb, http://conway.isri.cmu.edu/%7ejdh/MethodsF06/lec/probframes/prob-fr-3b.ppt

Recalling steps in problem frame modeling

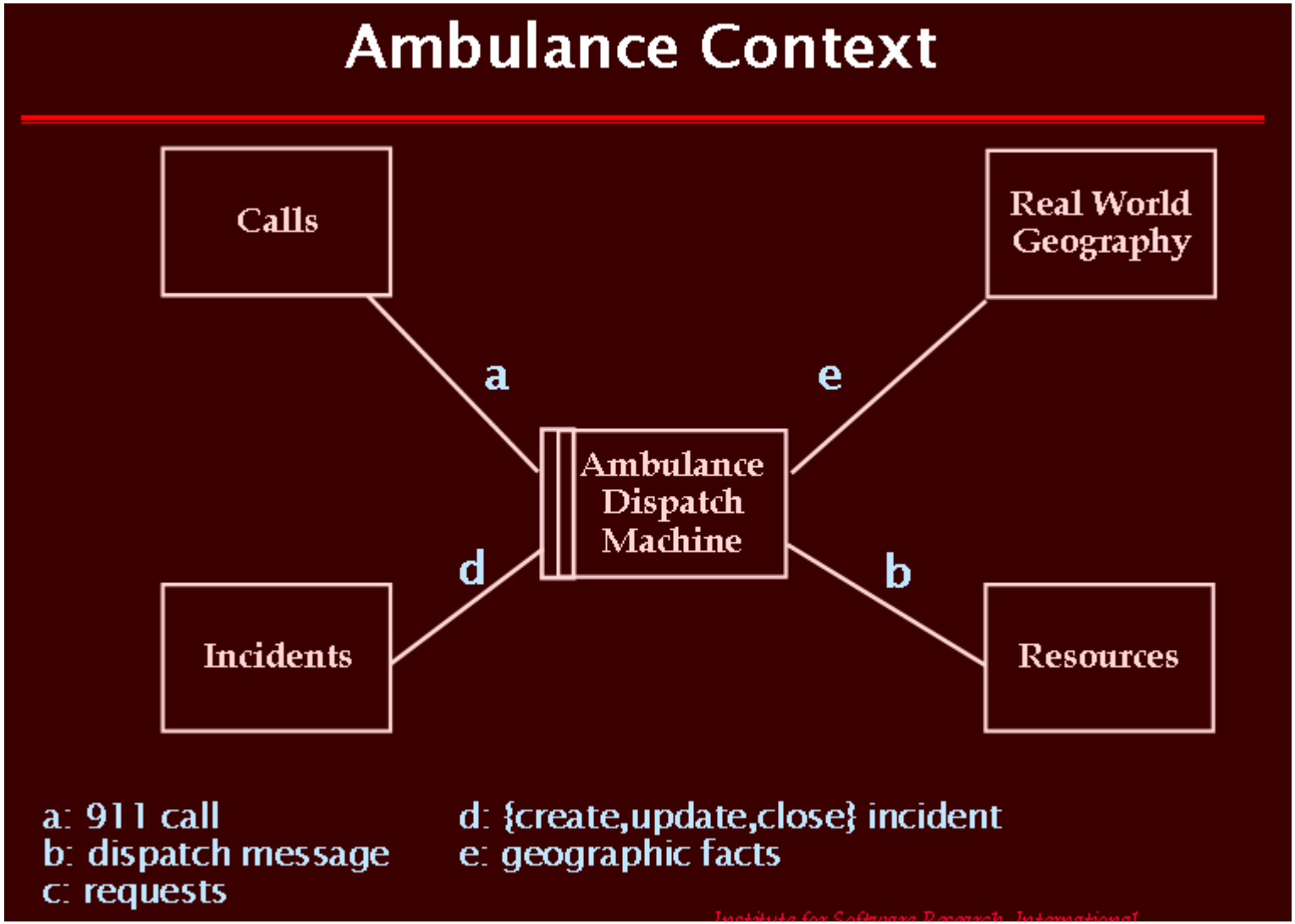


Source: Lecture note by Jim Herbsleb, http://conway.isri.cmu.edu/%7ejdh/MethodsF06/lec/probframes/prob-fr-1.pdf

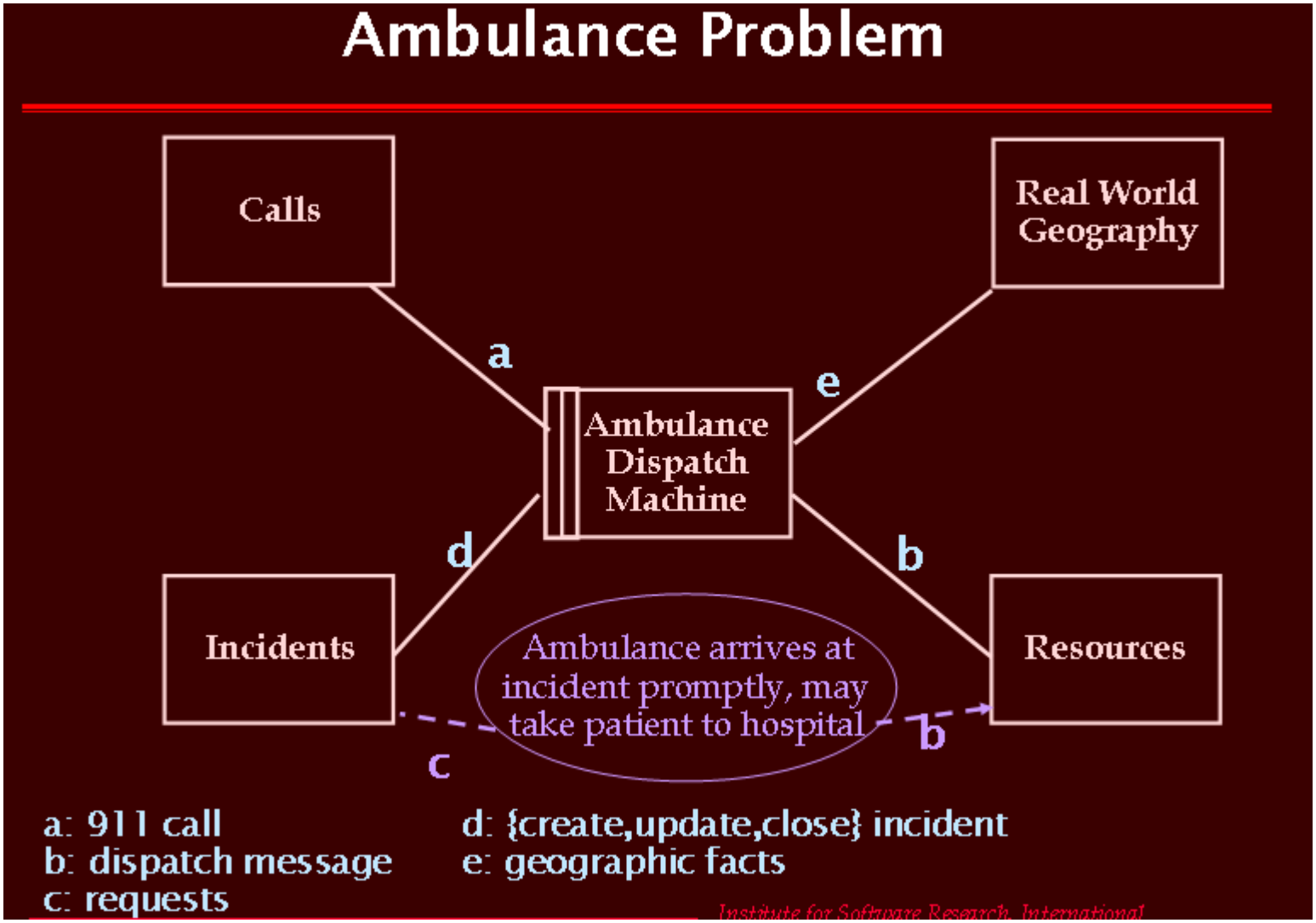
Problem Domains

- *Calls*: telephone calls from the public and doctors
- *Resources*: ambulances, personnel, special equipment
- But …
  - Calls do not correspond directly to incidents
  - Detailed knowledge of geography is required to interpret calls and to know which ambulance to send
- So add domains …
- *Incidents*: discrete events that require ambulance response
- *Geography*: Streets, addresses, hospital locations, etc

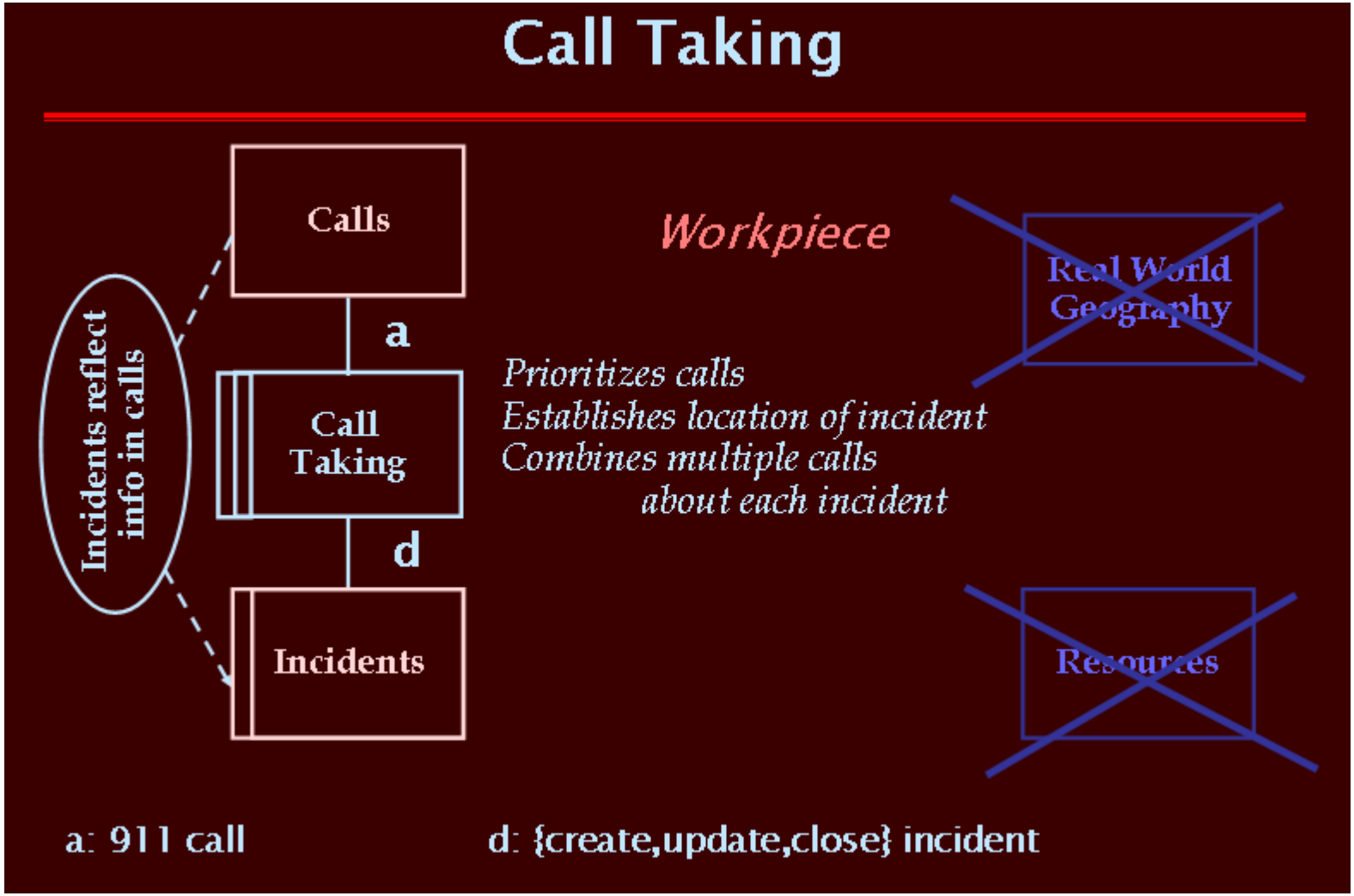
Source: Lecture note by Jim Herbsleb, http://conway.isri.cmu.edu/%7ejdh/MethodsF06/lec/probframes/prob-fr-3b.ppt



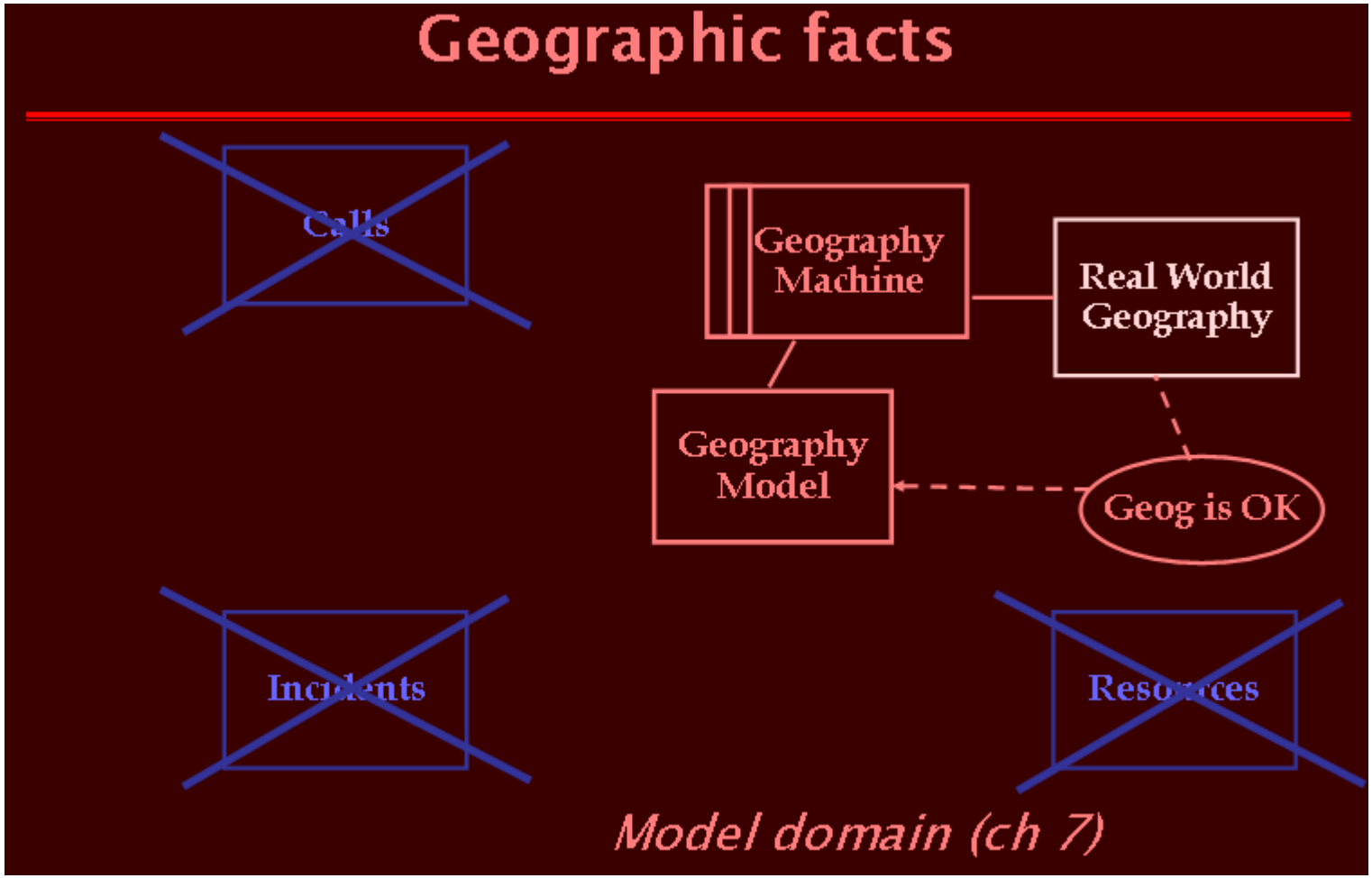
Source: Lecture note by Jim Herbsleb, http://conway.isri.cmu.edu/%7ejdh/MethodsF06/lec/probframes/prob-fr-3b.ppt



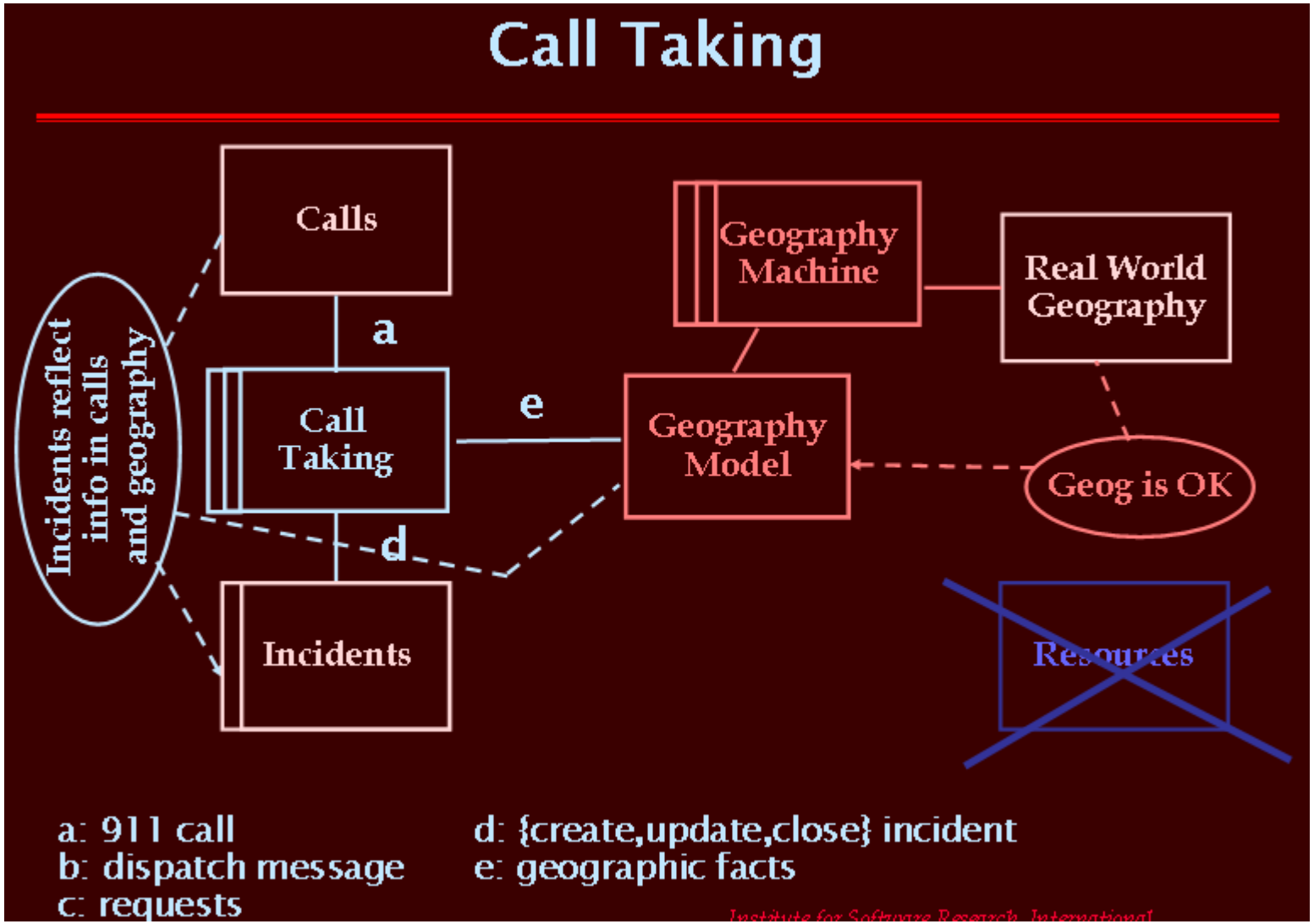
Source: Lecture note by Jim Herbsleb, http://conway.isri.cmu.edu/%7ejdh/MethodsF06/lec/probframes/prob-fr-3b.ppt



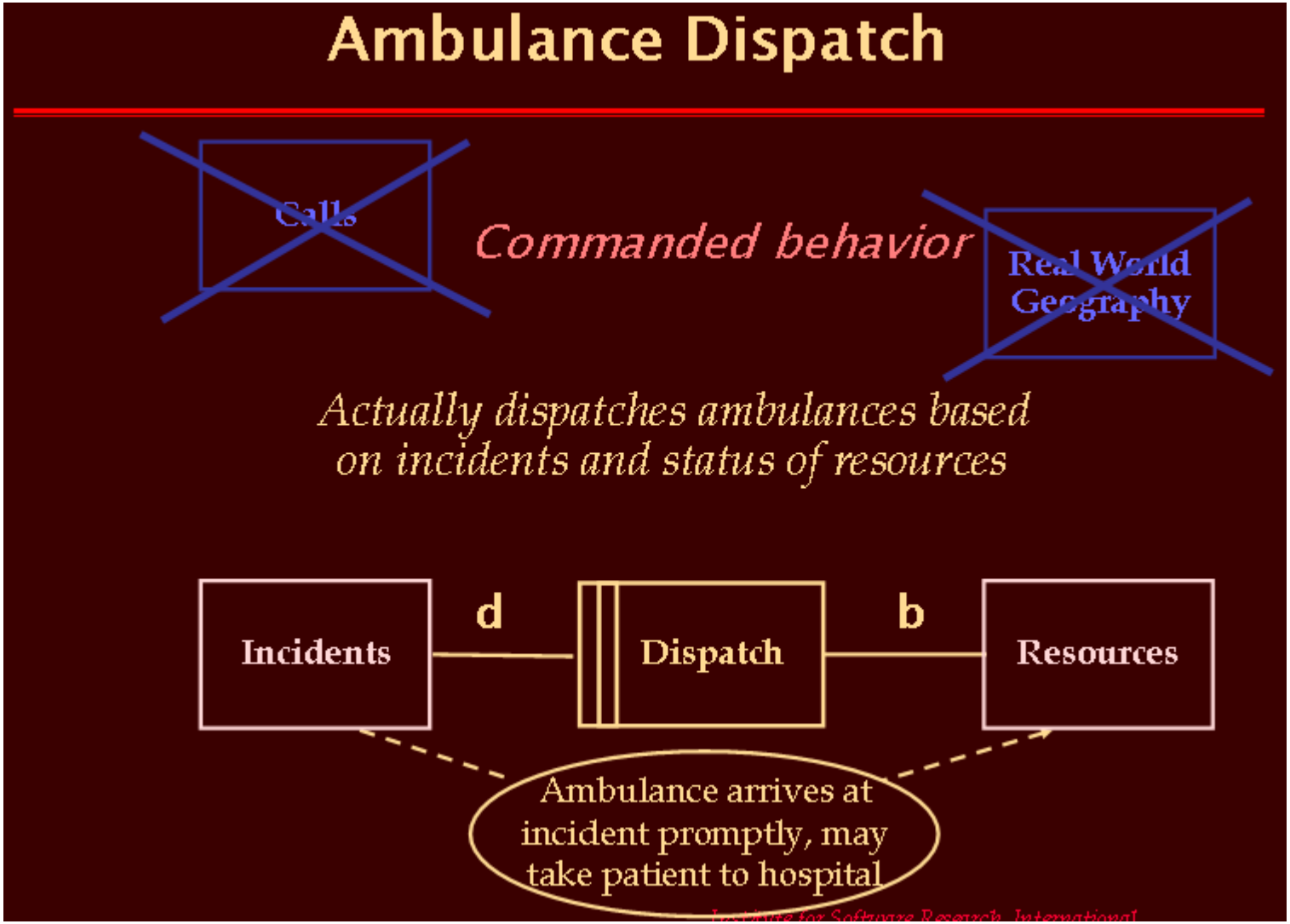
Source: Lecture note by Jim Herbsleb, http://conway.isri.cmu.edu/%7ejdh/MethodsF06/lec/probframes/prob-fr-3b.ppt



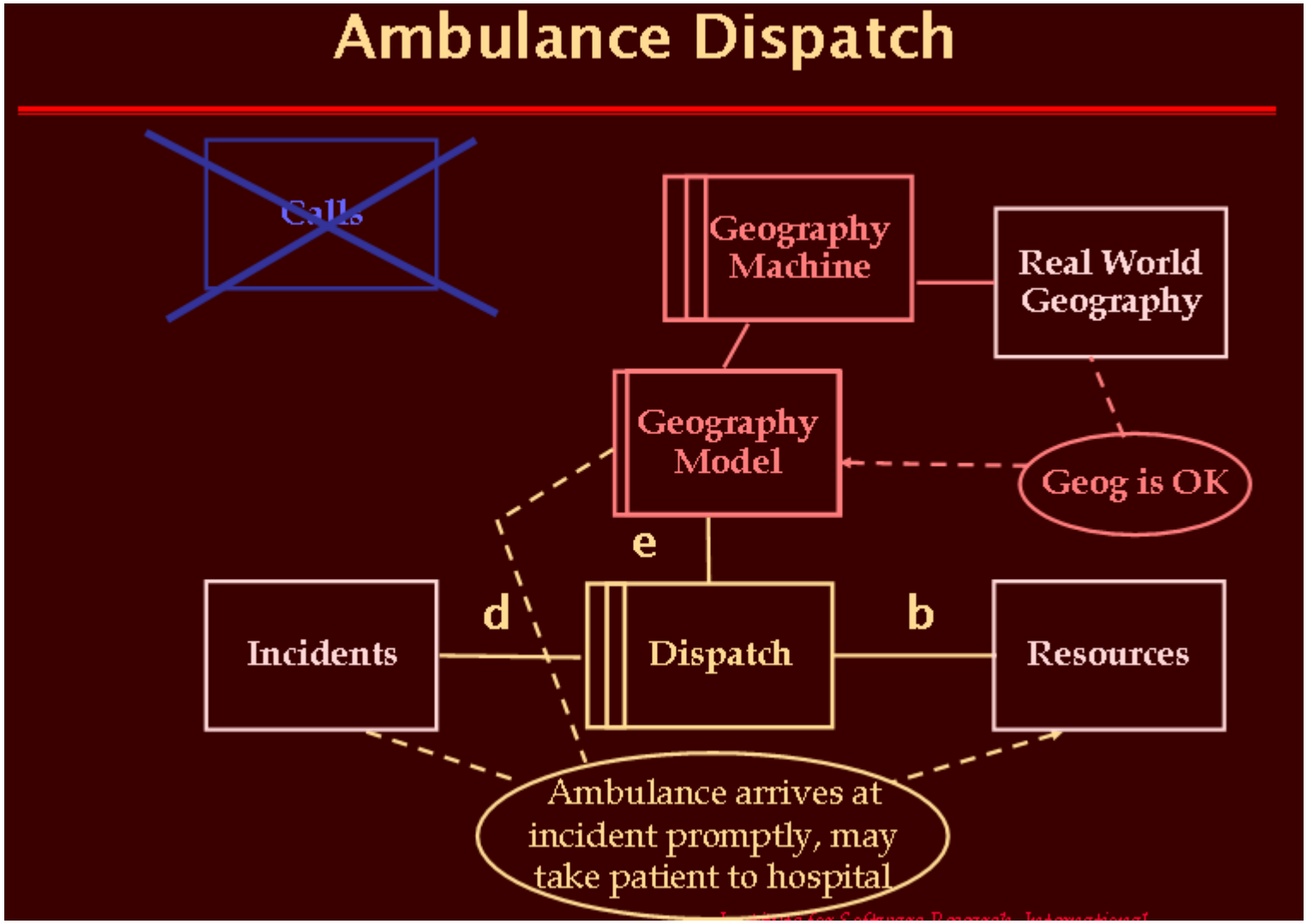
Source: Lecture note by Jim Herbsleb, http://conway.isri.cmu.edu/%7ejdh/MethodsF06/lec/probframes/prob-fr-3b.ppt



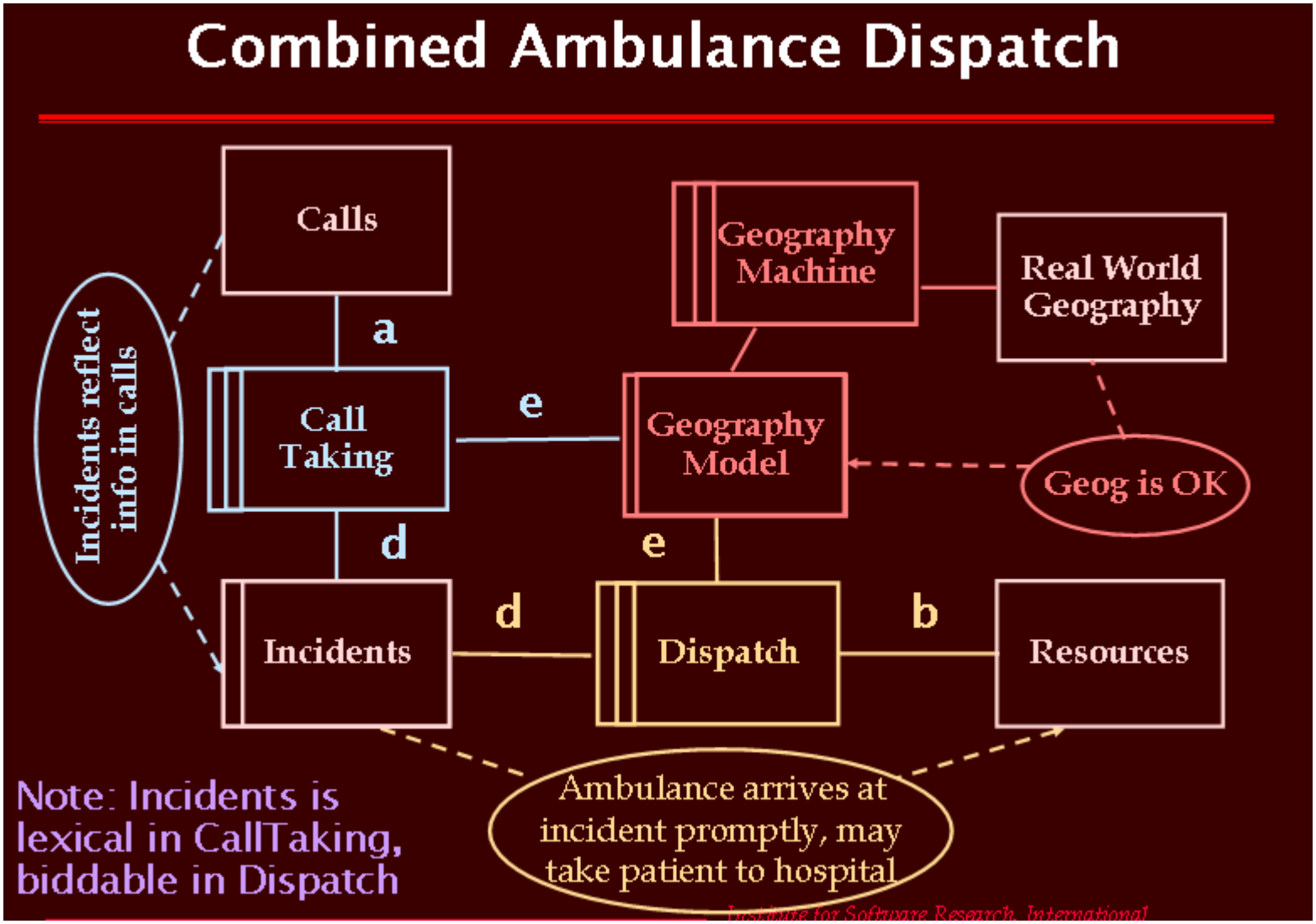
Source: Lecture note by Jim Herbsleb, http://conway.isri.cmu.edu/%7ejdh/MethodsF06/lec/probframes/prob-fr-3b.ppt



Source: Lecture note by Jim Herbsleb, http://conway.isri.cmu.edu/%7ejdh/MethodsF06/lec/probframes/prob-fr-3b.ppt



Source: Lecture note by Jim Herbsleb, http://conway.isri.cmu.edu/%7ejdh/MethodsF06/lec/probframes/prob-fr-3b.ppt



Source: Lecture note by Jim Herbsleb, http://conway.isri.cmu.edu/%7ejdh/MethodsF06/lec/probframes/prob-fr-3b.ppt

How to elicit?

Knowledge Acquisition: A Relative of Requirements Elicitation

- From AI, largely intended for acquiring expertise (e.g., of doctors, lawyers)
  - **practised by "knowledge engineer"**
  - Recall: requirements elicitation -> capturing "knowledge" of domain
- Use of mediating representations:
  - **help bridge the gap between the structure of expert' knowledge and formal, computer-based representations**
  - **(e.g., Text, Note, Diagram, Chart, Table, Frame, Rule, Semantic-Net)**
- Automatic KA techniques
  - infer new knowledge from past experience

    worksFor(bill, john)
    worksFor(maria, john)
    worksFor(george, john)
    ---
    forall x worksFor(x, john)

    For whom does Susan work ?
  - suggest refinement

    forall x, y (x <> y) -> worksFor(x, y)
  - detect inconsistencies

    forall x worksFor(x, john)
    worksFor(eve, maria)
- Issues recognized for KA
  - **novice K <> expert K** --> *diff. types of customers*
  - **experts may not want to tell** --> *"say-do" problem*
  - **expertise (experience) doesn't always translate into "rules"**

    --> *reqs. analyst: informal -> formal (ethnomethodology)*

How to elicit?

Data/Information Elicitation Techniques

- Sampling
- Questionnaires
- Interviewing
- Group Meetings
- Ethnomethodology
- Scenarios => 4

Requirements should contain *nothing but* information about the environment.

How to elicit?

Data/Information Elicitation Techniques

Sampling

- the process of systematically selecting representative elements of a population, often applied to documents ("hard" data, e.g., transaction log)
- useful as it can minimize costs/overhead during data gathering (only a portion, no direct involvement of customer)
- sampling tasks:
  - **data determination**

    E.g., in building/improving an AIM system
    - how much time/transaction (-> #machines, response time improvement)
    - how many errors before completion (-> UI design, robustness, help fns)
    - correlation between amount and time spent (-> max amt, accuracy assurance)
    - peak period, interval between transactions (-> performance improvement)
    - success/failure ratios (-> bad transaction types, time of day/week)
  - **population**

    E.g., transactions
    - transactions in 4 local branches for 1 week
  - **type determination**
    - **purposive sampling** choose population elements the analyst considers important with no regard to statistical issues (e.g., only high amount/frequent transactions)
    - **random sampling** every kth element
  - **sample size**

    E.g., consider 1/10th of all transactions (in 4 local branches for 1 week)

    the bigger the size, the higher the cost of sample collection, but higher confidence level

How to elicit?

Data/Information Elicitation Techniques

Questionnaires

- kinds of information sought: attitudes, beliefs, behavior
  - not normally found through sampling (hard data) or interviews

  But if not anonymous, customers may be reluctant to answer questions

  Have you used any meeting scheduler system before? Y N

  If yes, are you satisfied with it? 1 2 3 4 5
  If no, would you try a meeting scheduler system when available? Y N

  Would you encourage other people to use one? Y N

  How much time are you willing to spend in each session?

  5minutes< 5minutes< & <10 minutes 10 minutes< & <20 minutes

- avoid open questions
  (because answers to such questions are hard to correlate and interpret)

  Do you think a new meeting scheduler will succeed?
  Do you believe a mtg scheduler system should drastically change our daily lives?

- questionnaires should be short
  (otherwise, people may be reluctant to participate with busy schedule)

- administer the questionnaire using simple rules
  - scoring scheme: e.g., a range of from 1 to 5
  - group inter-related questions

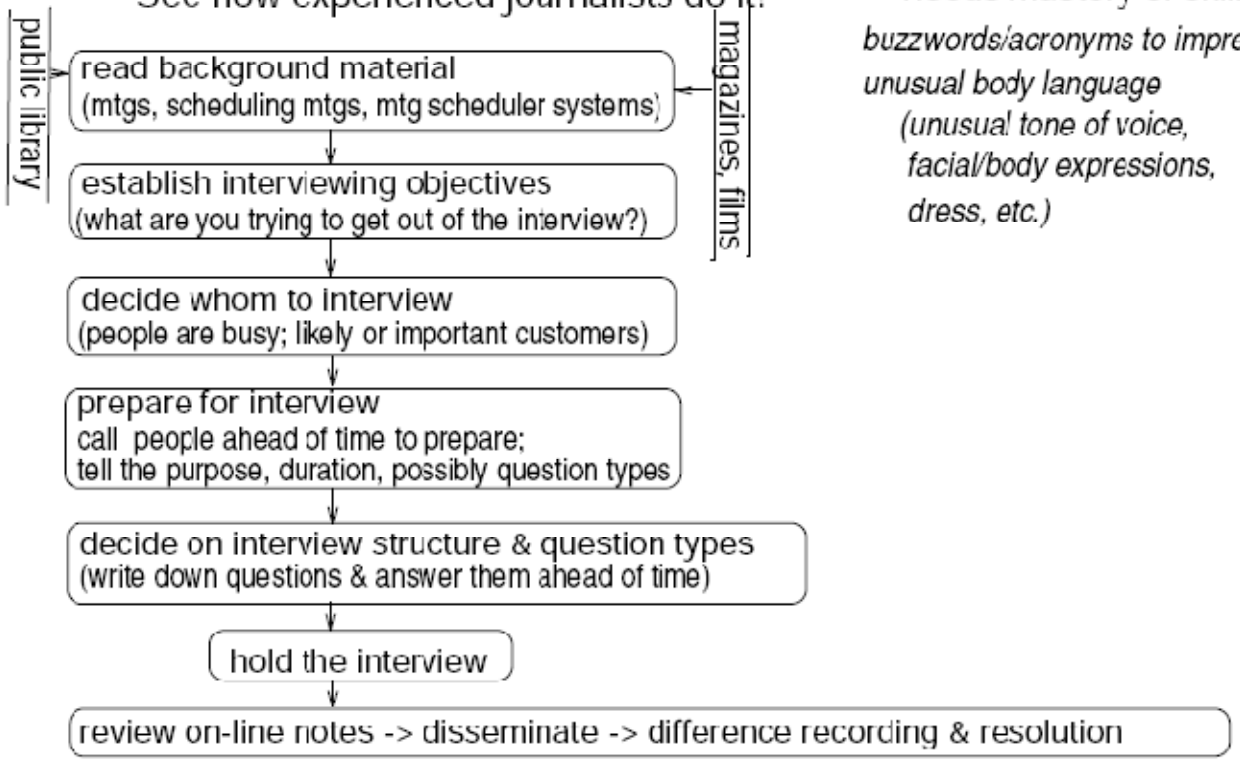
    E.g., Q 1 2 3 represent customer satisfaction with current systems
    Q 4 5 6 7 represent customer willingness to try a new one

How to elicit?

Data/Information Elicitation Techniques

Interviewing

- kinds of information sought:
  tacit knowledge as well as hard facts, opinions, feelings, goals
- dos: planning ahead of time
  See how experienced journalists do it!



- don'ts:
  needs mastery of skills
  buzzwords/acronyms to impress
  unusual body language
  (unusual tone of voice, facial/body expressions, dress, etc.)

often people can't articulate their perception or their needs;
often people are reluctant to reveal their thoughts

How to elicit?

Data/Information Elicitation Techniques

Group Meetings

- focus groups:
  - kind of group interview, often conducted in terms of "stimulus material" (videos, stories, ..
  - Success depends on the kinds of participants and moderator
- E.g., Joint Application Development (JAD)
  - Joint Requirements Planning (JRP)
    - usually for high-level managers;
    - identify and examine business goals, problems, critical success factors, strategic opportunities
  - Joint Application Design (JAD)
    - identify and examine the end users' needs
  - 4 tenets of JAD:
    - group dynamics
      - participants (developers, users/customers)
      - leader/moderator/facilitator
      - recorder/scribe
    - visual aids
      - E.g., calendars, participants, equipments, locations
    - organized, rational process
      - periodic, democratic, conflict accommodating
    - WYSIWIG documentation approach

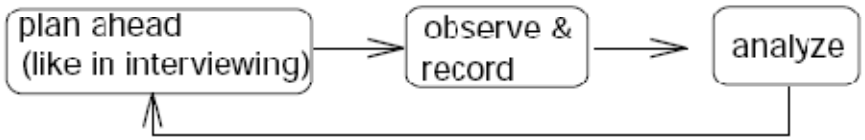
How to elicit?

Data/Information Elicitation Techniques

- the "say-do" problem: people know how to do things they normally don't describe (tacit knowledge); descriptions of such things may be highly inaccurate
  - experts may not want to tell ---> *"say-do" problem*

Ethnomethodology (People's methods)

- Sometimes, observation is the best way to understand how things are done
- (esp. where) social order is accomplished on a moment-to-moment basis
- So, OBSERVE in a NATURAL setting

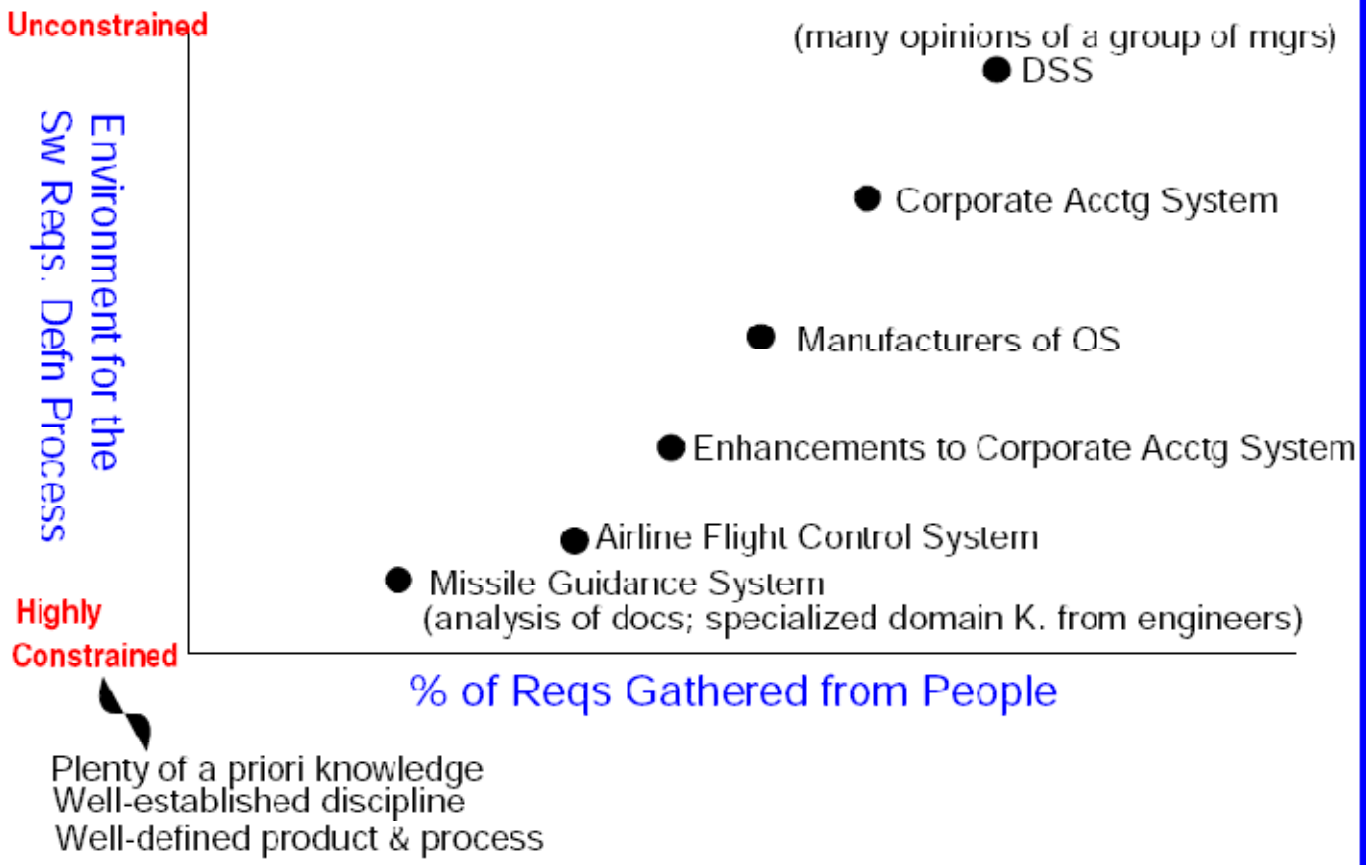


- e.g., stock brokerage (multiple phone calls, computer), IICI
- ethical, legal implications, if video-taping without notification
- observation not in a natural setting, if people are aware of being observed
  needs maximal natural setting, minimal interruption
- can be too time-consuming to analyze the recording
  gradual identification of critical tasks and focusing

Appendix II

Why is it difficult?

Sources of requirements [SEI]



Plenty of a priori knowledge
Well-established discipline
Well-defined product & process

Fraction of reqs. elicited from people increases as constraints on the sw reqs. process decrease

Lawrence Chung

What are the implications?

Cf. market-driven vs. customer-driven project

Desired Properties of the WRSPM Reference Model

❑ Some Preliminary: Church's higher-order logic notation

- A logic is called higher order if it allows sets to be quantified or if it allows sets to be elements of other sets.
- A higher-order predicate is a predicate that takes one or more other predicates as arguments.
- A higher-order predicate of order n takes one or more $n-1^{th}$-order predicates as arguments, where $n > 1$.
- A similar remark holds for higher-order functions.

- Examples:
  - The law of the excluded middle: $\forall P(P \vee \neg P)$
  - The law of non-contradiction: $\forall P \neg(P \wedge \neg P)$
  - The principle of bivalence: $\forall S\ \forall x\ (x \in S \vee x \notin S)$;
  - Leibniz's principle of equality: $a = b$ def= $\forall P\ [P(a) \equiv P(b)]$
  - Inequality: $\forall a\ \forall b\ (a \neq b \equiv \exists P\ (P(a) \wedge \neg P(b)))$
  - Metaclass Intension: $\forall x\ P_1\ P_2.\ P_1(x) \wedge \text{IS-A}(P_1, P_2) \rightarrow \text{IN}(x, P_2, s)$
  - Principle of mathematical induction (PMI): $\forall P(P(0) \wedge (\forall m(P(m) \rightarrow P(s(m)))) \rightarrow (\forall n P(n)))$, where $s^M(m) = m + 1$.
  - Reachability (FOL cannot express transitive closure)

    Let R(x, y) be the transition relation of the graph, then state u could reach v if:

    0 steps: $u = v$

    1 steps: $R(u, v)$

    2 steps: $\exists x_1(R(u, x_1) \wedge R(x_1, v))$

    3 steps: $\exists x_1\ \exists x_2(R(u, x_1) \wedge R(x_1, x_2) \wedge R(x_2, v))$

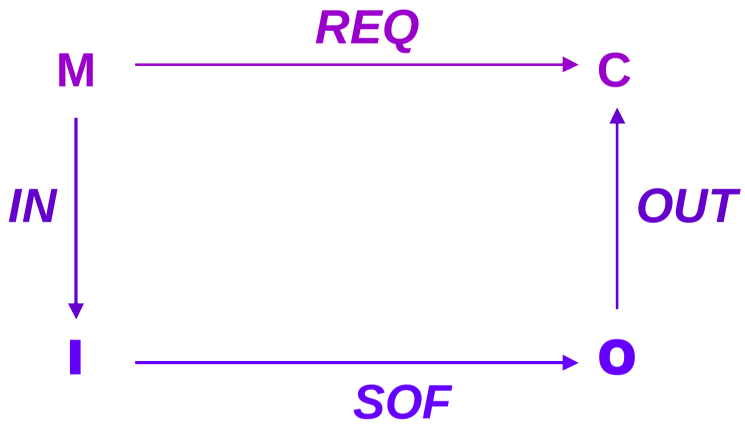
    ...

    But this is an infinitely long formula and hence not a WFF of predicate logic.

$\forall P(\forall x \forall y \forall z\ (P(x, x) \wedge (P(x, y) \wedge P(y, z) \rightarrow P(x, z)) \wedge (R(x, y) \rightarrow P(x, y))) \rightarrow P(u, v))$

Desired Properties of the WRSPM Reference Model

- **Some Preliminary: Church's higher-order logic notation**
  - ***The 4-variable model:***

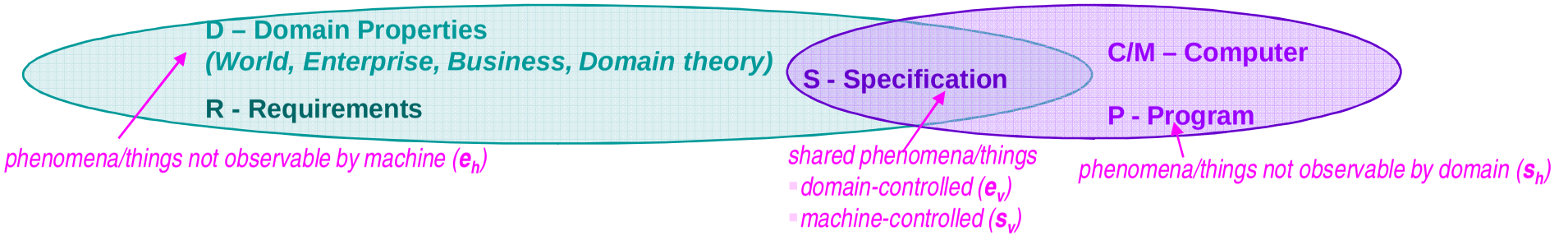


If *REQ* is a function specifying the abstract requirements (desired behaviour) of the software and
SOF is to be the concrete implementation (the program),
we want to know if a program exist that meets the requirements:

$$\exists\ SOF\ \forall\ m \in M\ (REQ(m) = OUT(SOF(IN(m))))$$

Desired Properties of the WRSPM Reference Model

[C. A. Gunter, E. L. Gunter, M. Jackson and P. Zave, A Reference Model for Requirements and Specifications. *IEEE Software* 17(3) 37-43. May/June 2000]



W restricts the actions that the environment can perform by restricting ***e*** or the relationship between ***e*** and s_v;
R describes more restrictions, saying which of all possible actions are desired;
S is expressed in the language common to the environment and system (states of **S** are in **W**).

If $e_h = \{x_1, \ldots, x_n\}$, then a formula $\forall\ e_h.\ \Phi$ means the same as $\forall\ x_1, \ldots, x_n.\ \Phi$

Adequacy	$\forall e\ s.\ W \wedge M \wedge P \Rightarrow R.$	(1)	The requirements allow all the events the environment performs (e_h, e_v) and all the events the system performs (s_v, s_h) that can happen simultaneously.
Consistency of W	$\exists e\ s.\ W.$	(2)	(same as $\exists\ e_h\ e_v\ s_v.\ W$; a non-triviality assumption)
Relative consistency	$\forall e_v.\ (\exists e_h\ s.\ W) \Rightarrow (\exists e_h\ s.\ W \wedge M \wedge P).$	(3)	Any choice of values for the environment variables visible to the system is consistent with M ∧ P if it is consistent with assumptions about the environment.
Adequacy wrt. S	$\forall e\ s.\ W \wedge S \Rightarrow R.$	(4)	if S properly takes W into account in saying what is needed to obtain R, and P is an implementation of S for M, then P implements R as desired.
Relative consistency for S	$\forall e_v.\ (\exists e_h\ s.\ W) \Rightarrow (\exists s.\ S) \wedge (\forall s.\ S \Rightarrow \exists e_h.\ W).$	(5)	
Relative consistency for $M \wedge P$ wrt. S	$\forall e.\ (\exists s.\ S) \Rightarrow (\exists s.\ M \wedge P) \wedge (\forall s.\ (M \wedge P) \Rightarrow S).$	(6)	(5), (6) => (3); (4), (5), (6) => (1) *Cf. in paper, (3) and (1) are switched*

if the software buyer is responsible for the environment-side proof obligations & the "seller" is responsible for the system-side obligations, then the buyer must satisfy Formulas 2, 4, and 5, and the seller must satisfy Formula 6.

More on the WRSPM Reference Model

Adequacy	$\forall e\ s.\ W \wedge M \wedge P \Rightarrow R.$	(1)	The requirements allow all the events the environment performs (*eh*, *ev*) and all the events the system performs (*sv*, *sh*) that can happen simultaneously.
Consistency of W	$\exists e\ s.\ W.$	(2)	(same as $\exists\ e_h\, e_v\, s_v.\ W$; a non-triviality assumption)
Relative consistency	$\forall e_v.\ (\exists e_h\ s.\ W) \Rightarrow (\exists e_h\ s.\ W \wedge M \wedge P).$	(3)	Any choice of values for the environment variables visible to the system is consistent with M ∧ P if it is consistent with assumptions about the environment.
Adequacy wrt. S	$\forall e\ s.\ W \wedge S \Rightarrow R.$	(4)	if S properly takes W into account in saying what is needed to obtain R, and P is an implementation of S for M, then P implements R as desired. (5), (6) => (3); (4), (5), (6) => (1) *Cf. in paper, (3) and (1) are switched*
Relative consistency for S	$\forall e_v.\ (\exists e_h\ s.\ W) \Rightarrow (\exists s.\ S) \wedge (\forall s.\ S \Rightarrow \exists e_h.\ W).$	(5)	
Relative consistency for M ∧ P wrt. S	$\forall e.\ (\exists s.\ S) \Rightarrow (\exists s.\ M \wedge P) \wedge (\forall s.\ (M \wedge P) \Rightarrow S).$	(6)	

$\exists\ e\ s.\ W \Rightarrow M \wedge P.$ ***Too weak for formula 3***: This says that there is some choice of the environment events that makes the system consistent with the environment. However, the environment might not use only this consistent set of events. This formula should hold, and it does immediately follow from the domain-knowledge consistency (Formula 2) and relative consistency (Formula 3).

$\forall e\ s.\ W \Rightarrow M \wedge P.$ ***Too strong for formula 3***, because it means that any choice of potential system behavior (*s*) that W accepts, the system to be built (M ∧ P) must also accept.

$\forall e.\ \exists s.\ W \Rightarrow M \wedge P.$ ***Too weak for formula 3***, because given an environment action, it lets the system do anything it chooses if there is a corresponding value for the system actions that invalidates the domain knowledge.

$\forall e.\ (\exists s.\ W) \Rightarrow (\exists s.\ W \wedge M \wedge P).$

Too strong for formula 3: The environment actions hidden from the system include reactions to the machine's behavior. The machine is not allowed to restrict any of the possible environmental reactions. It is consistent with the domain knowledge for the patient's heart to stop beating (e_v) without the nurse being warned (e_h)—this is the undesirable possibility that the program is supposed to prevent. However, this formula states that, if this can happen, then the program must allow it! ?

$\forall e_v.\ (\exists e_h\ s.\ W) \Rightarrow (\exists e_h\ s.\ W \wedge S).$ (7)

Too weak for formula 5: This is a direct consequence of Formlua 5. The added constraint in Formula 5 means that W must hold everywhere that S holds. The weaker constraint only requires that there is somewhere that they both hold. (If we let S = S_1 V S_2, where S_1 is a good spec satisfying formula 5 ad guarantees R and S_2 is a bad spec everywhere inconsistent with W, then S satisfies Formula 4 and the weaker Formula 7, but not Formula 5.)

$\forall e.\ (\exists s.\ W) \Rightarrow (\exists s.\ W \wedge S).$ (5') ***Too ??? for formula 5***

More on the WRSPM Reference Model

❑ Some Exercise

Adequacy	$\forall e\ s.\ W \wedge M \wedge P \Rightarrow R.$ (1)	The requirements allow all the events the environment performs (*eh*, *ev*) and all the events the system performs (*sv*, *sh*) that can happen simultaneously.
Consistency of W	$\exists e\ s.\ W.$ (2)	(same as $\exists\ e_h\ e_v\ s_v\ W$; a non-triviality assumption)
Relative consistency	$\forall e_v.\ (\exists e_h\ s.\ W) \Rightarrow (\exists e_h\ s.\ W \wedge M \wedge P).$ (3)	Any choice of values for the environment variables visible to the system is consistent with M ∧ P if it is consistent with assumptions about the environment.
Adequacy wrt. S	$\forall e\ s.\ W \wedge S \Rightarrow R.$ (4)	if S properly takes W into account in saying what is needed to obtain R, and P is an implementation of S for M, then P implements R as desired. (5), (6) => (3); (4), (5), (6) => (1) *Cf. in paper, (3) and (1) are switched*
Relative consistency for S	$\forall e_v.\ (\exists e_h\ s.\ W) \Rightarrow (\exists s.\ S) \wedge (\forall s.\ S \Rightarrow \exists e_h.\ W).$ (5)	
Relative consistency for M ∧ P wrt. S	$\forall e.\ (\exists s.\ S) \Rightarrow (\exists s.\ M \wedge P) \wedge (\forall s.\ (M \wedge P) \Rightarrow S).$ (6)	

D_1: There will always be a nurse close enough to hear the buzzer
D_2: The sound from the heart falling below a certain threshold indicates that heart has (is about to) stop
R_1: A warning system notifies the nurse if the patient's heartbeat stops

S_1: If the sound from the sensor falls below a certain threshold, the buzzer shall be actuated

C – with a microphone as a sensor and a buzzer as an actuator

e_h: the nurse and the heartbeat of the patient.
e_v: sounds from the patient's chest.
s_v: the buzzer at the nurse's station.
s_h: internal representation of data from the sensor.

D_3: There may be some nurse who is deaf
D_4: There may be some noise that distorts the sound from the heart.
D_5: There may be a buzzer on a visitor's cell phone.
D_6: The heart stops beating, but the sound is above the threshold.
D_7: The heart is beating, but the sound falls below the threshold.
D_8: The sound falls below the threshold, but the nurse is not notified.
D_9: The sound falls below the threshold, but the buzzer is not actuated.
D_{10}: The sound falls below the threshold, but the buzzer is not actuated.
D_{11}: The sound is above the threshold, but the nurse is notified otherwise.
D_{12}: The heart is beating, but the nurse is not notified.
R_3: A warning system notifies a nurse in a remote location if the heartbeat stops
R_4: An entertainment system plays music if the sound is above a certain threshold.

S_2: If the sound from the sensor falls below a certain threshold, the buzzer shall be turned off for 1 second.

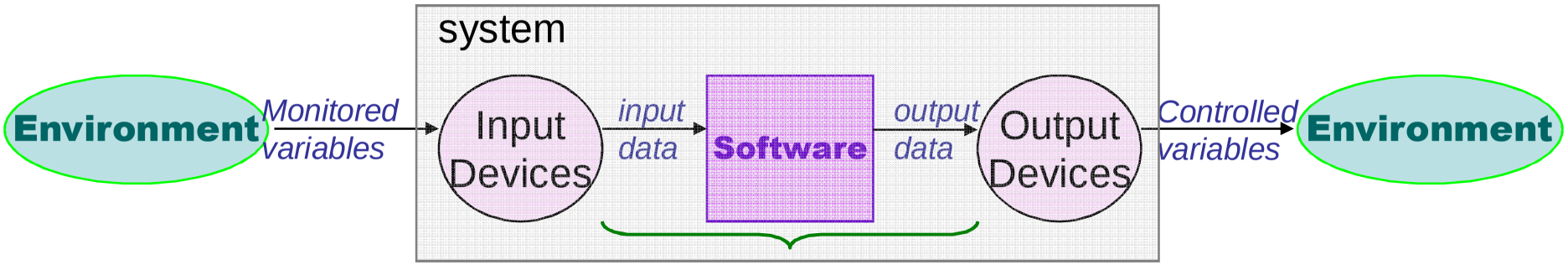
C_1 – with a malfunctioning microphone
C_2 - with a malfunctioning buzzer

$E_{h'}$: the nurse with amnesia and the heartbeat of the cat.
$E_{v'}$: sounds from the cat's chest.
$S_{v'}$: the buzzer on a visitor's cell phone.
$S_{h'}$: internal representation of data from the sensor.

Recall

The 4-Variable Model

(the functional documentation model)



S - Specification of software in terms of inputs & outputs

(possibly large in number, and in very complex relationships)

NAT(*m*, *c*): describes nature without making any assumptions about the system;

REQ(*m*, *c*): describes the desired system behavior;

IN(*m*, *i*): relates the monitored real-world values to their corresponding internal representation;

OUT(*o*, *c*): relates the software-generated outputs to external system-controlled values; and

SOF(*i*, *o*): relates program inputs to program outputs.

Any issues?

Three major proof obligations:

Feasibility: $\forall m.\ (\exists c.\ \mathrm{NAT}(m,\ c)) \Rightarrow (\exists c.\ \mathrm{NAT}(m,\ c) \wedge \mathrm{REQ}(m,\ c))$

IN must handle all cases possible under NAT: $\forall m.\ (\exists c.\ \mathrm{NAT}(m,\ c)) \Rightarrow (\exists i.\ \mathrm{IN}(m\ ,i))$

Acceptability: $\forall m\ i\ o\ c.\ \mathrm{NAT}(m,\ c) \wedge \mathrm{IN}(m,\ i) \wedge \mathrm{SOF}(i,\ o) \wedge \mathrm{OUT}(o,\ c) \Rightarrow \mathrm{REQ}(m,\ c)$

WRSPM Model vs. 4-Variable Model

Relationship between the two models:

4-Var	WRSMP
NAT	W
REQ	R + S
SOF	P
IN + OUT	M

4-Var	WRSMP
i, o	s_h
m	e_v
c	s_v
	e_h

NAT and REQ are more restricted than W and R, because they can only make assertions about those phenomena of the environment that are shared with the system.

IN and OUT together correspond to the M, except that they are more restricted, being limited to the special purposes of sensing and actuating.

Adequacy	$\forall e\ s.\ W \wedge M \wedge P \Rightarrow R.$	(1)	The requirements allow all the events the environment performs (*eh*, *ev*) and all the events the system performs (*sv*, *sh*) that can happen simultaneously.
Consistency of W	$\exists e\ s.\ W.$	(2)	(same as $\exists\ e_h\ e_v\ s_v.\ W$; a non-triviality assumption)
Relative consistency	$\forall e_v.\ (\exists e_h\ s.\ W) \Rightarrow (\exists e_h\ s.\ W \wedge M \wedge P).$	(3)	Any choice of values for the environment variables visible to the system is consistent with M ∧ P if it is consistent with assumptions about the environment.
Adequacy wrt. S	$\forall e\ s.\ W \wedge S \Rightarrow R.$	(4)	if S properly takes W into account in saying what is needed to obtain R, and P is an implementation of S for M, then P implements R as desired.
Relative consistency for S	$\forall e_v.\ (\exists e_h\ s.\ W) \Rightarrow (\exists s.\ S) \wedge (\forall s.\ S \Rightarrow \exists e_h.\ W).$	(5)	(5), (6) => (3); (4), (5), (6) => (1) *Cf. in paper, (3) and (1) are switched*
Relative consistency for M ∧ P wrt. S	$\forall e.\ (\exists s.\ S) \Rightarrow (\exists s.\ M \wedge P) \wedge (\forall s.\ (M \wedge P) \Rightarrow S).$	(6)	

Formula 1 is the same as acceptability.
Formula 2 translates to $\exists m\ c.$ NAT(m, c), which is not explicit because it is assumed to be true by construction.
Formula 3 translates to $\forall m.\ (\exists c.\ \text{NAT}(m, c)) \Rightarrow (\exists i\ o\ c.\ \text{NAT}(m, c) \wedge \text{IN}(m, i) \wedge \text{SOF}(i, o) \wedge \text{OUT}(o, c))$. It also subsumes the second (unnamed) obligation.
Formulas 1 and 3 imply Feasibility obligation: $\forall e_v.\ (\exists e_h\ s_v.\ W) \Rightarrow (\exists e_h\ s_v.\ W \wedge R)$.

Three major proof obligations:

Feasibility: $\forall m.\ (\exists c.\ \text{NAT}(m, c)) \Rightarrow (\exists c.\ \text{NAT}(m, c) \wedge \text{REQ}(m, c))$

IN must handle all cases possible under NAT: $\forall m.\ (\exists c.\ \text{NAT}(m, c)) \Rightarrow (\exists i.\ \text{IN}(m, i))$

Acceptability: $\forall m\ i\ o\ c.\ \text{NAT}(m, c) \wedge \text{IN}(m, i) \wedge \text{SOF}(i, o) \wedge \text{OUT}(o, c) \Rightarrow \text{REQ}(m, c)$

WRSPM Model vs. 4-Variable Model

Deficiencies of the 4-variable model: *(Formula 3 is missing)*

Although widely accepted, the three proof obligations in the 4-variable model are not sufficient.

Consider the following example, where all the variables are real-valued functions of time:

$$\begin{aligned}
\text{Nat}: &\quad (\forall t.c(t) > 0) \wedge (\forall t.m(t) < 0) \\
\text{Req}: &\quad \forall t.c(t+3) = -m(t) \\
\text{In}: &\quad \forall t.i(t+1) = m(t) \\
\text{SOF}: &\quad \forall t.o(t+1) = i(t) \\
\text{Out}: &\quad \forall t.c(t+1) = o(t)
\end{aligned}$$

Each predicate is internally consistent.

All the predicates besides NAT are readily implementable, because all establish relationships between their inputs at one time and their outputs at a later time.

The predicates satisfy all the proof obligations of the Functional Documentation Model, yet they are not realizable, because

$$\neg(\exists m\ i\ o\ c.\ \mathbf{NAT}(m, c) \wedge \mathbf{IN}(m, i) \wedge \mathbf{SOF}(i, o) \wedge \mathbf{OUT}(o, c)).$$

This unrealizability would have been revealed by

Formula 3 translates to $\forall m.\ (\exists c.\ \text{NAT}(m, c)) \Rightarrow (\exists i\ o\ c.\ \text{NAT}(m, c) \wedge \text{IN}(m, i) \wedge \text{SOF}(i, o) \wedge \text{OUT}(o, c))$.

If the program flipped the sign of its input to get the delayed output, all would be well.

The acceptability obligation is satisfiable only because the antecedent of its implication is always false.

$$\forall m\ i\ o\ c.\ \text{NAT}(m, c) \wedge \text{IN}(m, i) \wedge \text{SOF}(i, o) \wedge \text{OUT}(o, c) \Rightarrow \text{REQ}(m, c)$$