Problem Frames

A Tutorial

Syed Ahsan Fahmi
ISER Lab, KAIST
24 September 2009

Some of the contents of this tutorial is taken from lecture slides by Mr. Michel Jackson

Overview

- Introduction
- Elementary problem frames
- Some example problems
- Discussion on students requirements
- Problem frames and concerns
- References

Problem classes

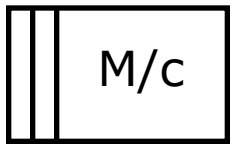
- Are all software development problems are the same?
  - Of course not
  - But…….May have similar sub-problems that fall into same few classes
  - …Example:
    - Detecting and reporting failure of analog devices in patient monitoring system
      - VS
    - Detecting and reporting malfunction in home heating control system

Introduction — 1

- We will focus on problems
  - Describing, classifying, structuring problems
  - Recognising concerns and difficulties ...
  - ... but never exploring solutions
- The approach is informal
  - Appropriate, because problems are in the world, and most of the world is informal
- Presentation is not rigorous
  - Much notation and terminology assumed or implied
- No particular development process is assumed

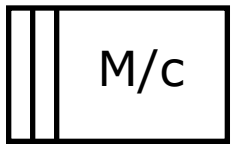
Introduction — 2

- Developing software is building a machine ...



Introduction — 2

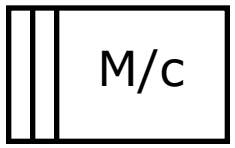
- Developing software is building a machine ...



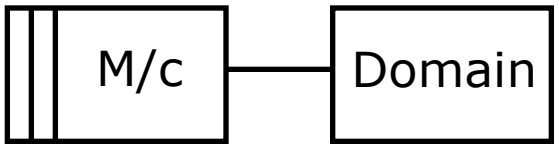
- One problem, one machine
- The machine is a general-purpose computer ...
  ... specialised by software
- The machine may be distributed
- One computer may support many machines
- Problem decomposition gives many subproblems ...
  ... and so many machines

Introduction — 2

- Developing software is building a machine ...

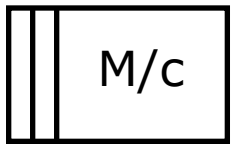


- ... to solve a problem in a given domain (a part of the world) ...

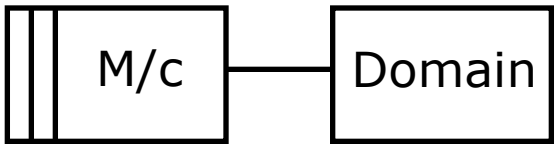


Introduction — 2

- Developing software is building a machine ...



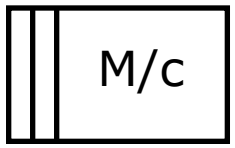
- ... to solve a problem in a given domain (a part of the world) ...



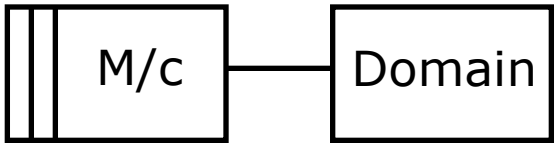
- The machine and the problem domain interact ...
  ... at an interface of shared phenomena (events, states, etc etc)
- Usually we need to structure the problem domain ...
  ... and to structure the problem into subproblems
- The machine in one subproblem may be a part of the problem domain in another subproblem

Introduction — 2

- Developing software is building a machine ...



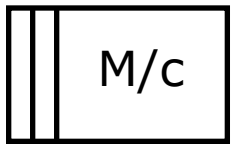
- ... to solve a problem in a given domain (a part of the world) ...



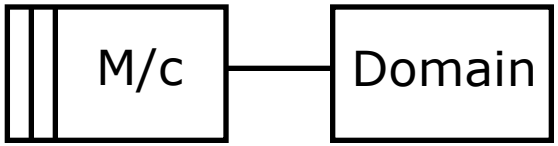
- ... to meet a customer's needs (the requirement)

Introduction — 2

- Developing software is building a machine ...



- ... to solve a problem in a given domain (a part of the world) ...

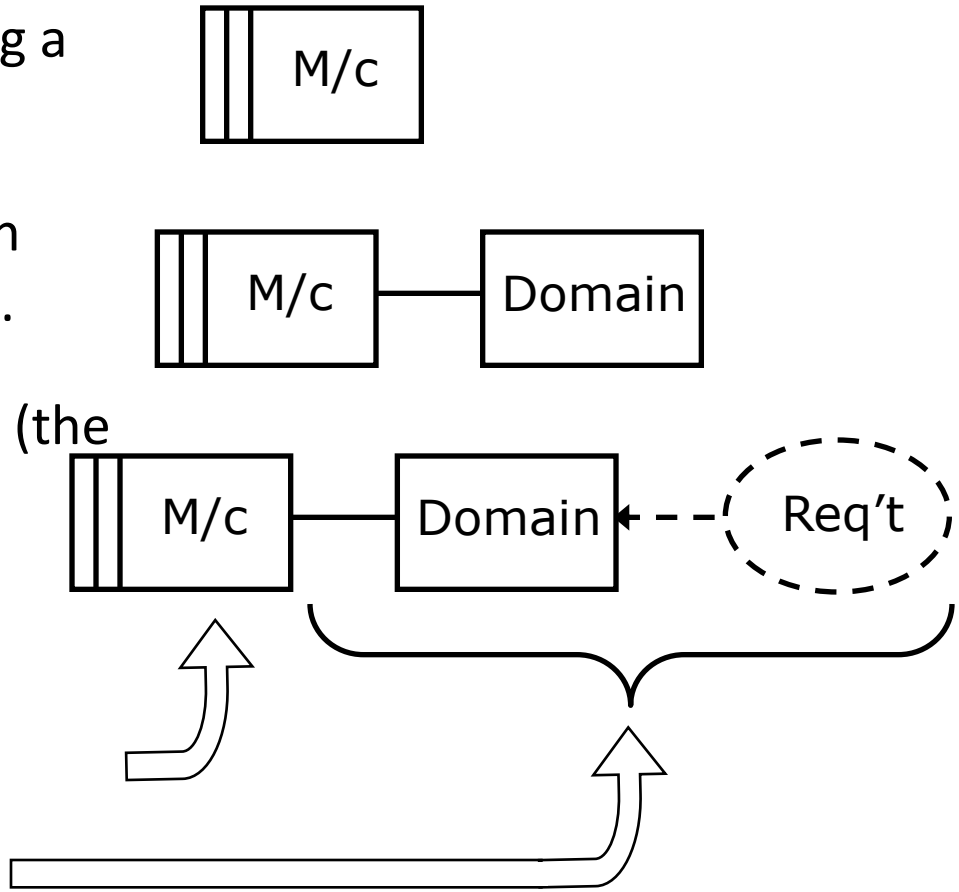


- ... to meet a customer's needs (the requirement)

- The customer's requirement is for some effect (or property or behaviour) in the problem domain
- ←- - - means that the requirement adds a constraint to the domain's intrinsic properties or behaviour

Introduction — 2

- Developing software is building a machine ...
- ... to solve a problem in a given domain (a part of the world) ...
- ... to meet a customer's needs (the requirement)
- The machine is the solution
- The problem is here:
  it is not inside the machine



Basic Problem Frames

- Required (Simple) behavior
  - Commanded behavior
- (Simple) Information display
  - Simple workpieces
  - Transformation

Phenomena- 1

- Six kinds of phenomena
  - First 3 are kinds of individuals
  - Last 3 are relations among individuals
- What is individual?
  - Something that can be named
  - Reliably distinguished from other individuals

Phenomena- 2

- 3 kinds of individuals
  - *Events*: an individual happening, taking place in a particular point of time
    - Events can never occur simultaneously
  - *Entities*: an individual that persists and can change its properties and states over time
    - A *patient* in the patient monitoring system is an *entity*
  - *Values*: individuals that exits outside time and space, and not subject to change
    - Ex. A *period* (the maximum between successive checks) in patient monitoring system

Phenomena- 3

- Relations among individuals
  - *States*: relation among entities and values which can change over time
    - Ex. Failed (AnalogDevice123)
  - *Truths*: a relation that cannot possible change over time
    - Ex. LengthOf(“ABCDE”,5)
  - *Roles*: a relation between an event and individuals in a particular way
    - A role can be thought of “argument” of an event

Phenomena- 4

- Casual phenomena:
  - Events, role, states relating entities..
  - Directly caused or controlled by some domain
- Symbolic phenomena:
  - Values, truths, states related only values…
  - Symbolizes other phenomena

Domain types

- Casual domains: properties include predictable casual relationships among casual phenomena
  - Ex. Mechanical and electrical equipments of a lift
- Biddable domains: consists of people.
- Lexical domains: physical representation of data
  - Combines casual and symbolic phenomena

Problem frames

Required (Simple) Behavior

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.

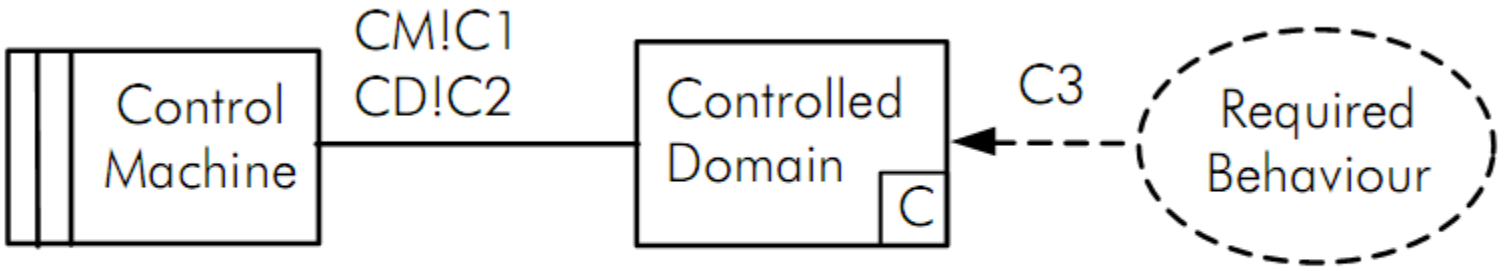
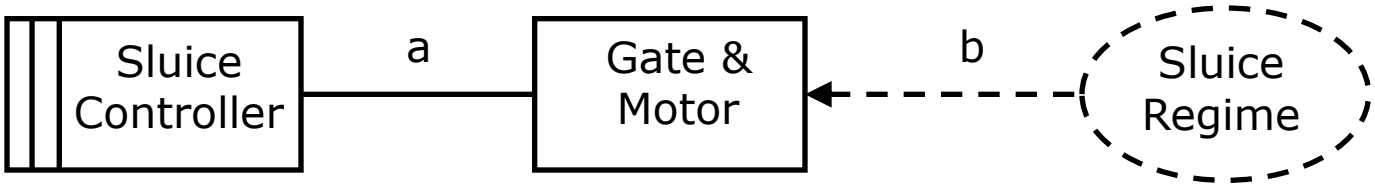


Figure 1. Problem Frame Diagram: Simple Behaviour

Example — 1

- A computer system is needed to control the sluice gate in a simple irrigation system. The gate must be held in the fully *Open* position for 10 minutes in every three hours and otherwise kept in the fully *Shut* position.
- The gate is opened and closed by vertical screws driven by a small motor controlled by *Clockwise*, *Anti-clockwise*, *On* and *Off* pulses. There are sensors at the *Top* and *Bottom* of the gate travel; at the top it is fully open, at the bottom it is fully shut. The connection to the computer consists of four pulse lines for motor control and two status lines for the gate sensors.

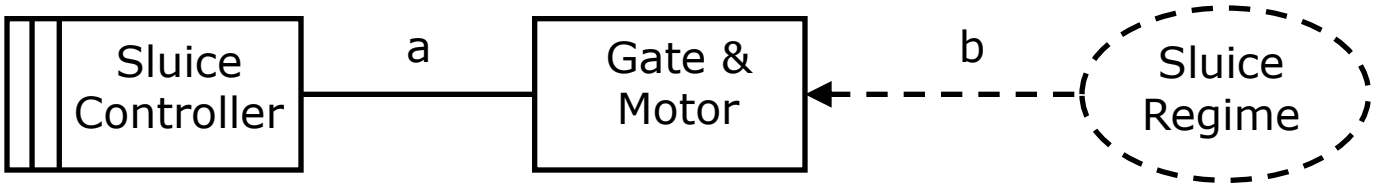
Sluice Gate Control



a: SC!{Clockw,Anti,On,Off}
GM!{Top,Bottom}

b: {Open,Shut}

Sluice Gate Control

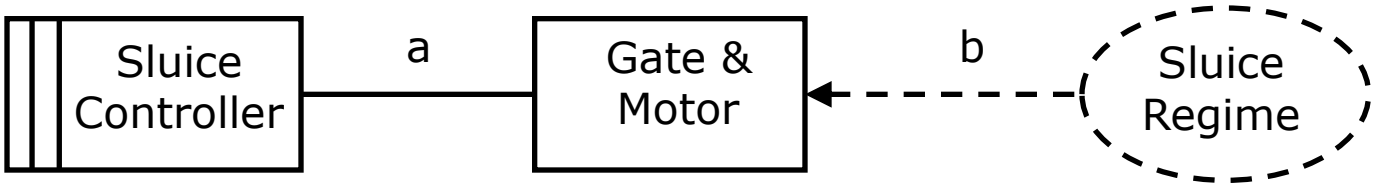


a: SC!{Clockw,Anti,On,Off}
GM!{Top,Bottom}

b: {Open,Shut}

- Interface annotations (a) show which domain controls the phenomena of each class
  - SC controls Clockw,Anti,On and Off events
  - GM controls Top and Bottom states

Sluice Gate Control



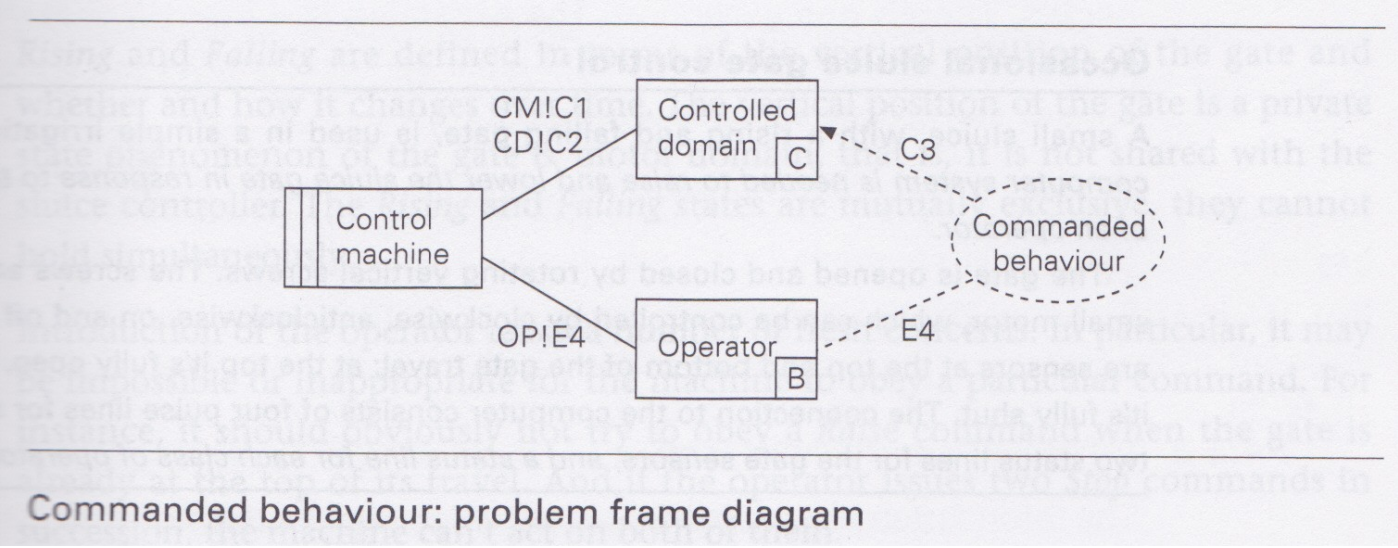
a: SC!{Clockw,Anti,On,Off}
GM!{Top,Bottom}

b: {Open,Shut}

- Reference annotations (b) show which domain phenomena are mentioned in the requirement
  - The customer cares only about whether the sluice gate is Open or Shut
- {Open, Shut} ✵ {Top, Bottom}

Commanded behavior

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.



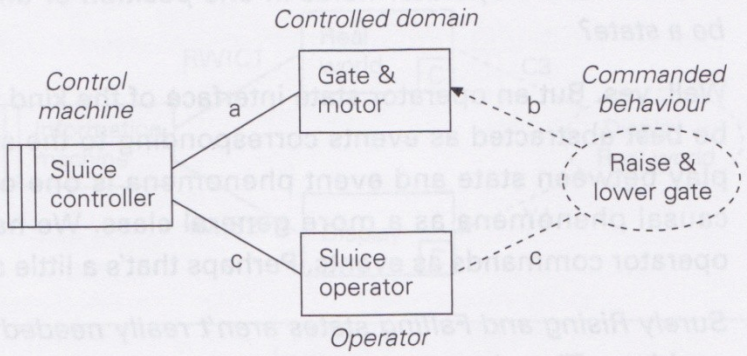
Commanded behaviour: problem frame diagram

Example 2

Occasional sluice gate control

A small sluice, with a rising and falling gate, is used in a simple irrigation system. *A computer system is needed to raise and lower the sluice gate in response to the commands of an operator.*

The gate is opened and closed by rotating vertical screws. The screws are driven by a small motor, which can be controlled by clockwise, anticlockwise, on and off pulses. There are sensors at the top and bottom of the gate travel; at the top it's fully open, at the bottom it's fully shut. The connection to the computer consists of four pulse lines for motor control, two status lines for the gate sensors, *and a status line for each class of operator command.*



Occasional sluice gate: fitted to commanded behaviour frame

Simple Information Display

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.

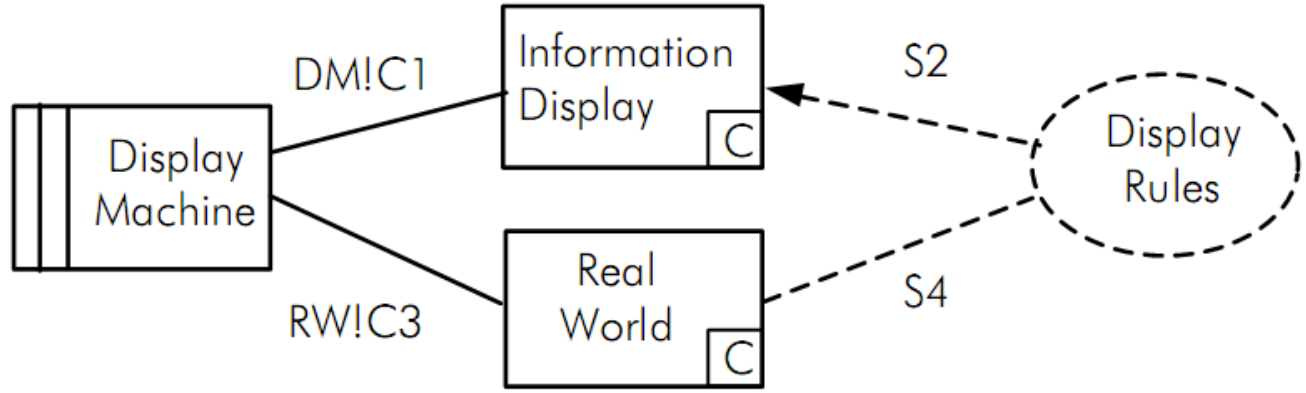
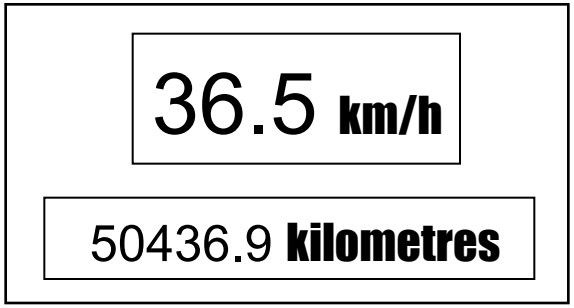


Figure 3. Problem Frame Diagram: Simple Information Display

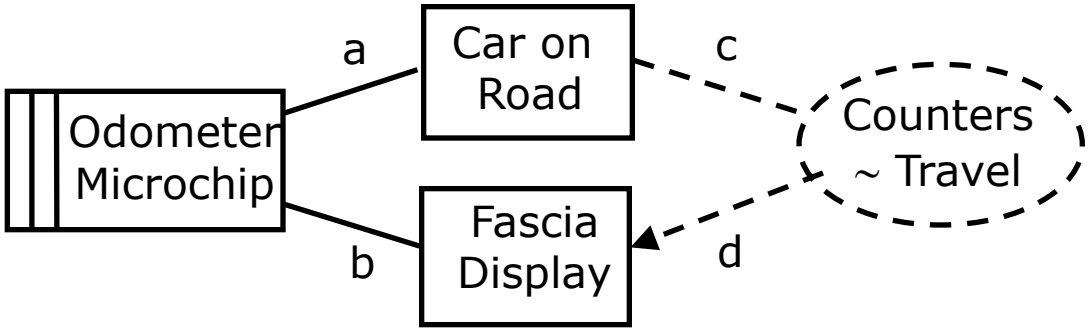
Examples — 3

- A microchip computer is required to control a digital electronic speedometer and odometer in a car:



- One of the car's rear wheels generates a pulse on each rotation. The computer can detect these pulses and must use them to set the current speed and total number of miles travelled in the two visible counters on the car fascia. The underlying registers of the counters are shared by the computer and the visible display.

Odometer Display



a: CR!{WheelPulse}
b: OM!{Inc/Dec Speed/Dist}

c: {Speed,CumDist}
d:{SpeedCount,DistCount}

Simple Workpieces

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.

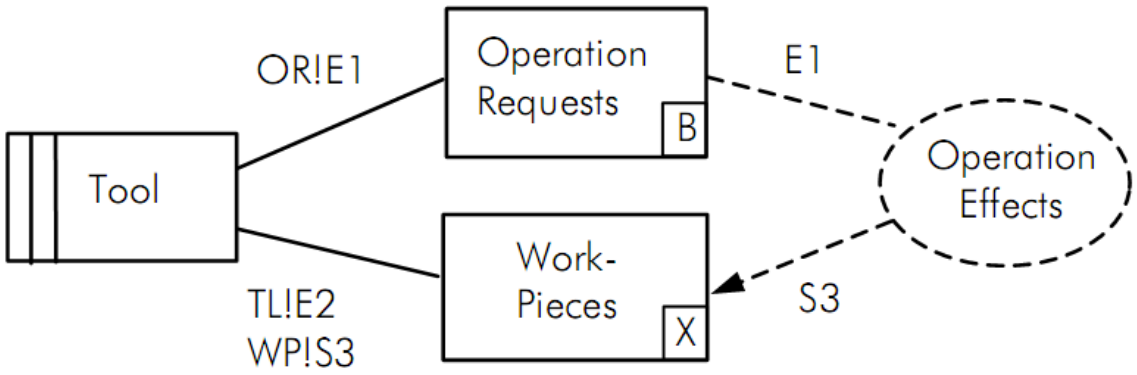
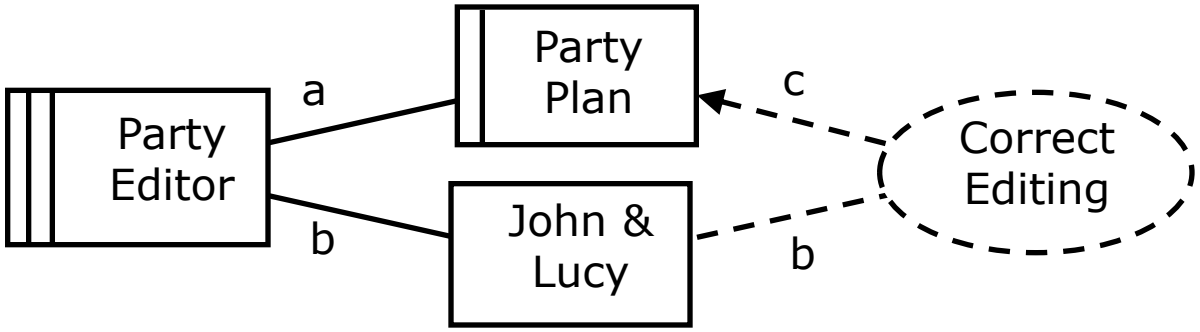


Figure 4. Problem Frame Diagram: Simple Workpieces

Example — 4

- Lucy and John give many parties, to which they invite many guests. They want a simple editor to maintain the information, which they call their Party Plan.
- The Party Plan is a list of parties, a list of guests, and a note of who's invited to each party.
- The editor will accept command-line text input, in a (very old-fashioned) DOS or Unix style.
- We are not at all concerned with presenting or printing the information — just with creating and editing it.

Party Plan Editing



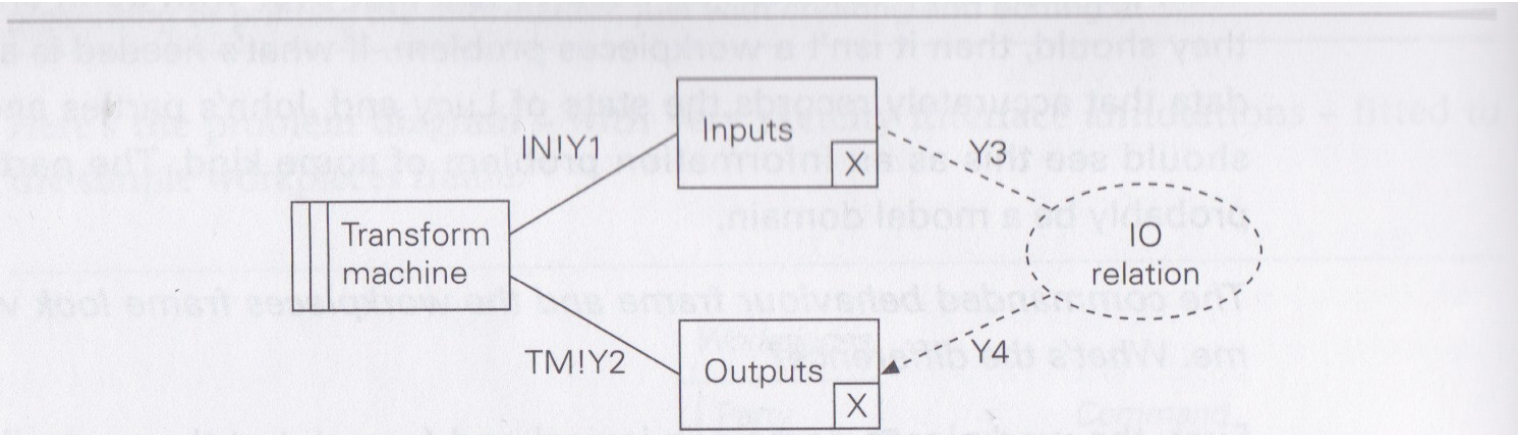
a: PE!{PlanOperations}
PP!{PlanStates}

b: JL!{Commands}

c: {PlanEffects}

Transformation

There are some given computer-readable input files whose data must be transformed to give certain required output files. The output data must be in a particular format, and it must be derived from the input data according to certain rules. The problem is to build a machine that will produce the required outputs from the inputs.



Transformation: problem frame diagram

Example- 5

Mailfiles analysis

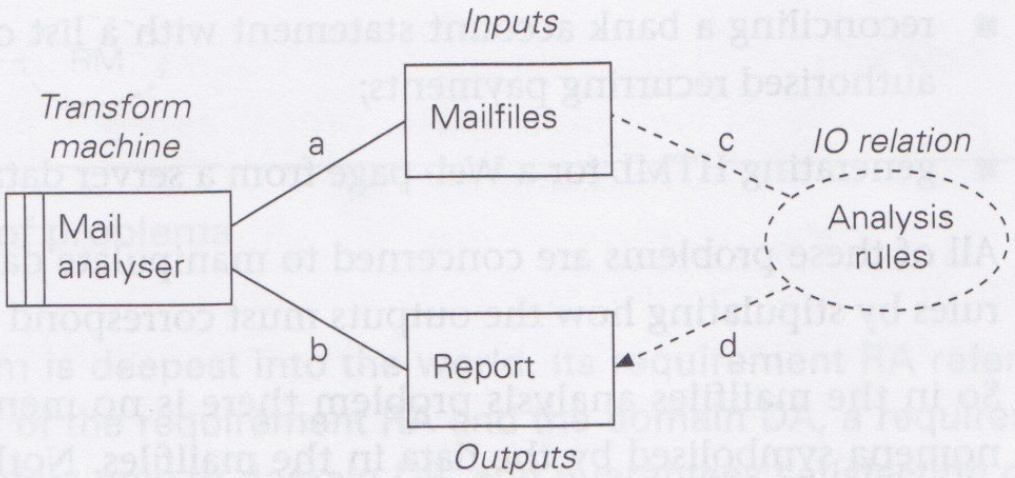
Fred has decided to write a program to analyse some patterns in his email. He is interested in the average number of messages he receives and sends in a week, the average and maximum message length, and similar things. After some thought he has worked out that he wants a report that looks like this:

Name	Days	#In	Max.Lth	Avg.Lth	#Out	Max.Lth	Avg.Lth
Albert	124	19	52136	6027	17	21941	2123
Anna	92	31	13249	1736	37	34763	2918
.....	...	...	.....	.....	...	.....	.....

Fred's mailfiles analysis report

The report contains a line for each of his correspondents. The line shows the correspondent's name, how many days the report covers, the number of messages received from the correspondent and their maximum and average lengths, and the same information for the messages sent to the correspondent by Fred.

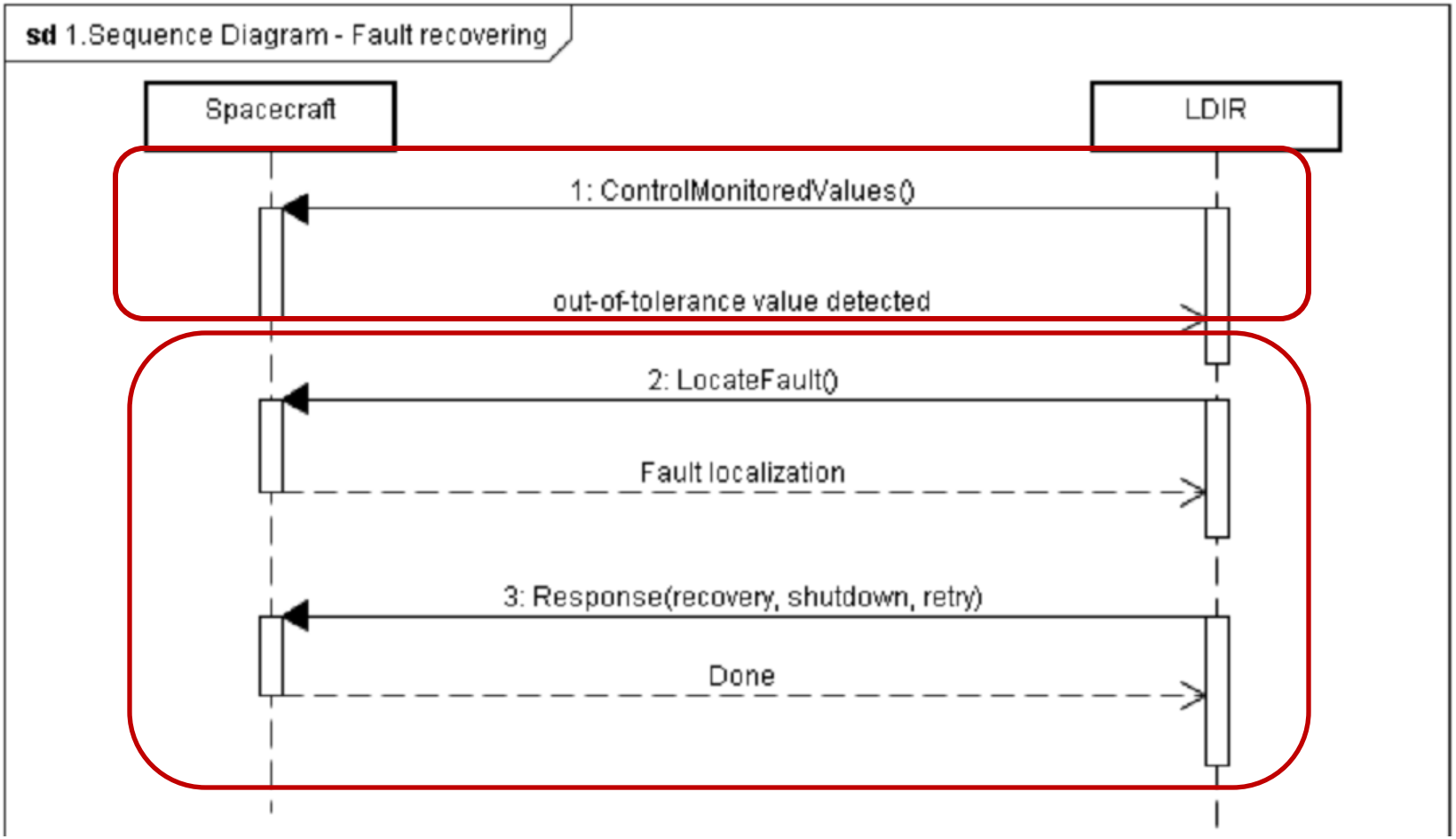
Example- 5



a: MF! {MsgDir, File, Line, Char} [Y1]
b: MA! {ReportLine, Char} [Y2]
c: MF! {Msg, From, To, Date, Length} [Y3]
d: RP! {LineData} [Y4]

Mailfiles analysis: fitted to transformation frame

Requirements from teams



Requirements from teams

Use Case Name: Isolate

Actors:FDIR (primary), Devices, Critical Devices (secondary)

Summary: The FDIR isolate the devices affected by the defective ones and takes control of the completion of the ongoing process.

Preconditions: The FDIR has identified the faulty devices and the interacting devices and the FDIR can be provided at the device layer

Description

Normal course: The interacting devices need to be maintained

1. The FDIR interrupts connection between the defective devices and its related ones
2. The FDIR identifies the critical devices and takes their functionement under control

Alternative Course: **Necessary to stop the interacting device for whatsoever purpose**

1. The FDIR interrupts the connection between the defective devices and their related ones
2. The FDIR shut down the interacting devices

Post conditions: The devices concerned by the fault are under the control of the FDIR or shut down

Priority: High (according to the type of fault detected)

Frequency of use: Random

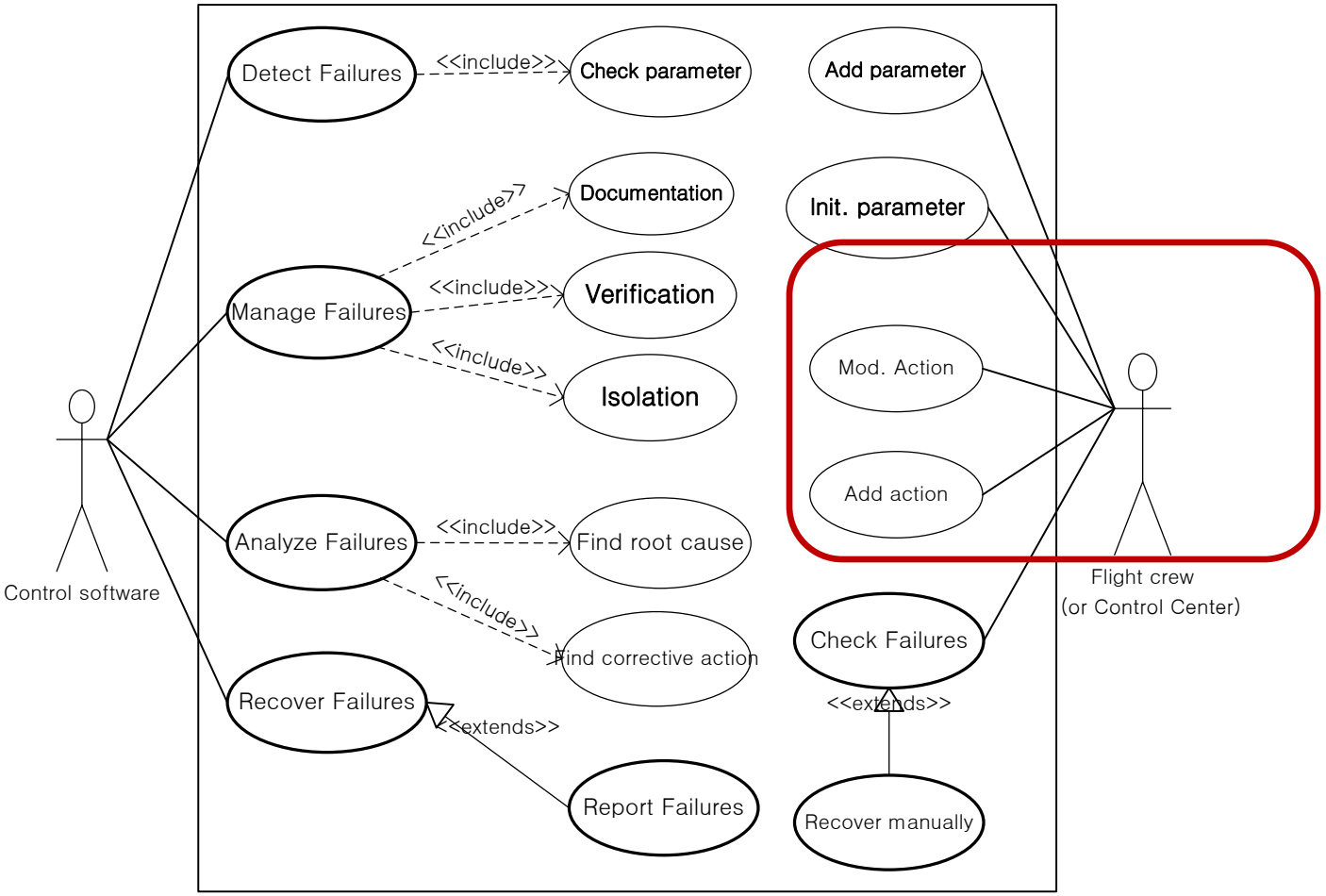
Special Requirements (Nun-functional)

Availability: The function should be always available

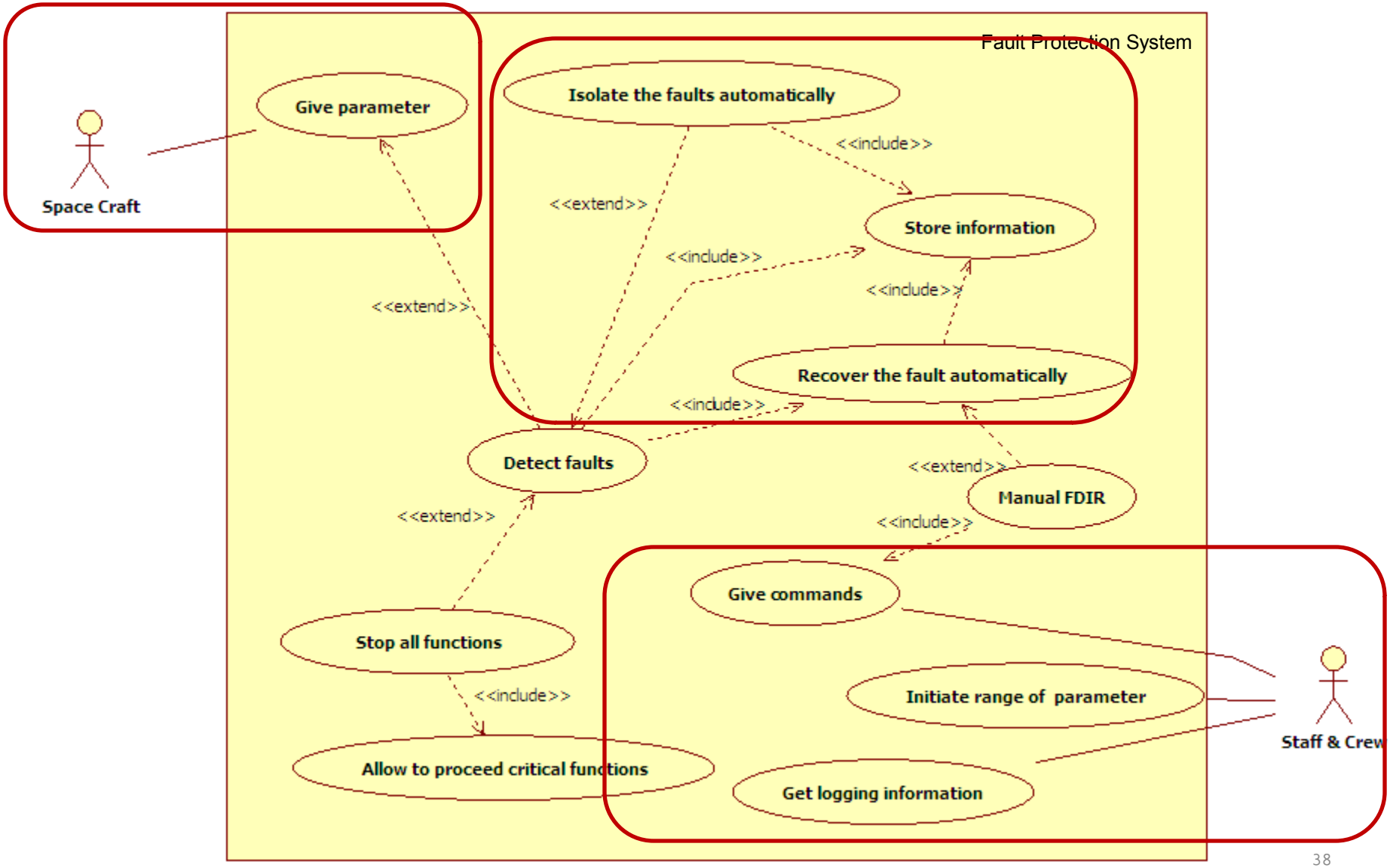
Response time: To be adjusted to safety critical quality standards

The isolation should allow emergency mechanical control

Requirements from teams



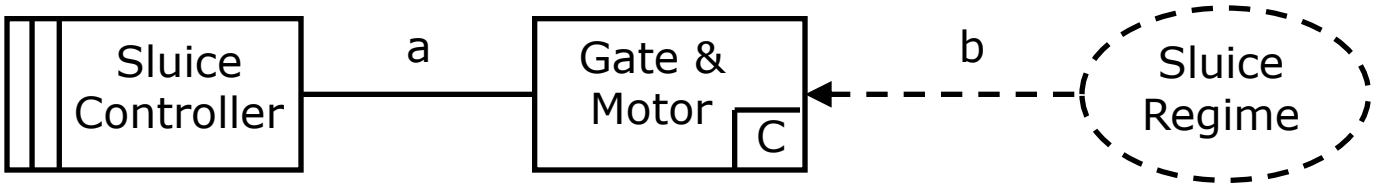
Requirements from teams



Problem Frames and Concerns — 1

- We have seen instances of four *problem frames*:
  - *Simple Behaviour* : Sluice Gate
  - *Simple Information* : Odometer Display
  - *Workpieces* : Party Plan Editing
- Each frame characterises a problem class
  - The problem is always to specify a suitable machine
- Frames differ in ...
  - ... types and configuration of domain parts
  - ... frame concern: what's needed to solve the problem
  - ... descriptions necessary to address frame concern

Sluice Gate Control

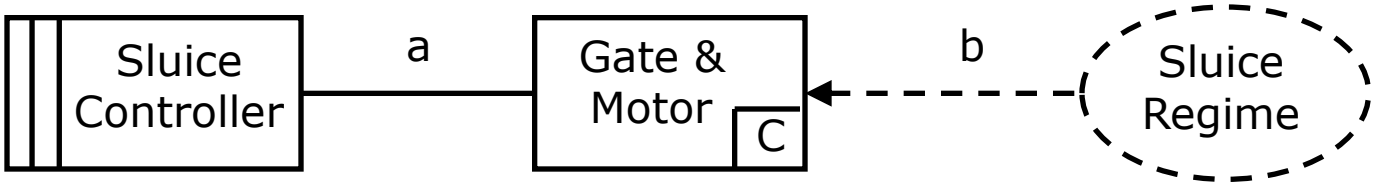


a: SC!{Clockw,Anti,On,Off}
GM!{Top,Bottom}

b: {Open,Shut}

- *Simple Behaviour Problem*
  - The *Controlled Domain* is a Causal domain
    - May be passive or weakly active or strongly active
  - The frame concern:
    - Find and exploit a control law to use phenomena (a) to satisfy the requirement in terms of phenomena (b)
    - Is there such a law?
    - Does *Control Machine* have enough timely information?

Problem Frames and Concerns — 2

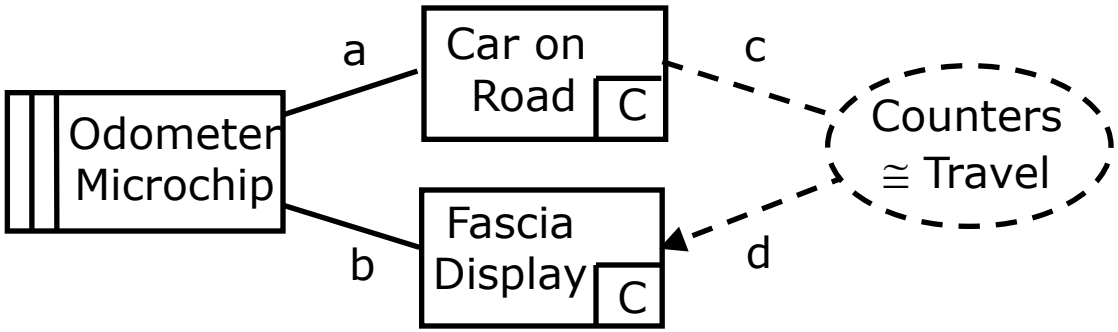


a: SC!{Clockw,Anti,On,Off}
GM!{Top,Bottom}

b: {Open,Shut}

- R: Requirements
  - What the customer *wants* to be true in terms of (b)
- D: Domain Properties
  - What we *know* to be true in the domain
- S: Specification
  - How we *want* the machine to behave at interface (a)
- General solution argument: S,D ↖ R
  - Similar arguments are needed for all frames

Odometer Display

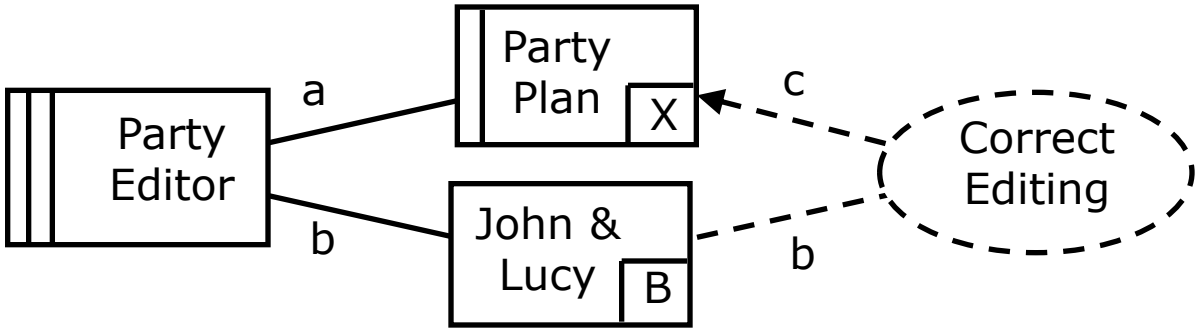


a: CR!{WheelPulse}
b: OM!{Inc/Dec Speed/Dist}

c: {Speed,CumDist}
d:{SpeedCount,DistCount}

- *Simple Information Problem*
  - The *Real World* and *Display Domain* are Causal domains
    - The *Real World* is autonomous
  - The frame concern:
    - Find and exploit domain properties by which the *Information Machine* can infer phenomena (c) from (a)

Party Plan Editing



a: PE!{PlanOperations}
PP!{PlanStates}

b: JL!{Commands}

c: {PlanEffects}

- *Workpieces Problem*
  - The *Workpieces* is a Lexical domain
    - Lexical domains are always passive
  - The *Users* are a Biddable domain
  - The frame concern:
    - Generate operations (a) on Workpieces to give
    - correct effects (c) of commands at (b)

Problem Frames and Concerns — 3

- Each problem frame has its characteristic concern
- Some concerns are common ...
  - ... to several frames, or ...
  - ... to domains of particular kinds
- Some examples:
  - *Untimely* phenomena in an information problem
  - *Reliability* of a causal domain in a behaviour problem
  - *Initial* states of machine and problem domain
  - Potential for *breakage* of a causal domain
  - *Identification* of individuals in multiple domain

References

- [Jac99] Michael Jackson, “Problem analysis using small problem frames”, Proc. WOFACS’98, Special Issue of the South African Computer Journal, 22, pp. 47-60, March 1999.
- [Jac00] Michael Jackson, Problem analysis and structure, Engineering Theories of Software Construction, T. Hoare, et al. eds: Proc. NATO Summer School, pp. 3-20, 2000.
- [Jac01] Michael Jackson, Problem Frames: Analysing and Structuring Software Development Problems, Addison-Wesley, 2001.
- [Jac05] Michael Jackson, “Problem frames and software engineering”, Information and Software Technology, Special Issue: 1st Int Workshop on Advances and Applications of Problem Frames, K. Cox, et al. eds, Vol. 47 No. 14, pp. 903-912, Nov. 2005.
- http://people.csail.mit.edu/dnj/teaching/6898/lecture-notes/session8/slides/mj-problem-frames.pdf
- http://www.jacksonworkbench.co.uk/stevefergspages/pfa/index.html