A Tutorial on
Software Development
Problem Frames
Part 1

BCS RESG Problem Frames Day
Open University
10 May 2006

Michael Jackson
jacksonma@acm.org

What Is Our Subject?

- Our subject is
  - An intellectual framework for thinking about problems and solutions in software engineering
  - A way of characterising problems and the concerns they raise for software engineers
  - An evolving repertoire of ways of understanding development techniques and difficulties
- Our subject is
  - Not a calculus or a formalism
  - Not a development method or process
  - Not a prescription for success in every problem
  - Not a complete prescription for any problem

A Tutorial In Two Parts

10.00-11.15 Part 1

Software development problems
Where is the problem?
Solving a problem
Problems and subproblems

11.30-12. 45 Part 2

Subproblem concerns
Subproblem composition
Composition concerns
Normal and radical design
Discussion

Additional Slides: Problem Frames Bibliography

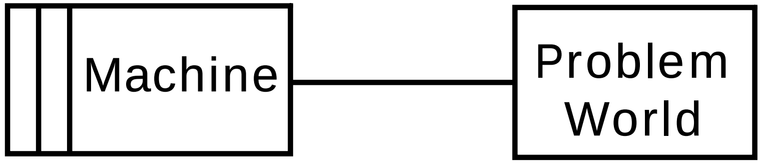
Software Development Problems

- Controlling traffic lights
- Supporting the administration of a library
- Controlling use of a car park
- Invoicing electricity consumers
- Monitoring patients in an intensive care unit
- Supporting web-based retail operations
- Controlling a lift
- Managing accounts in a bank
- Providing a tool for word processing
- Managing production in a factory
- Central locking in a car
- … …

A Problem Has a Problem World

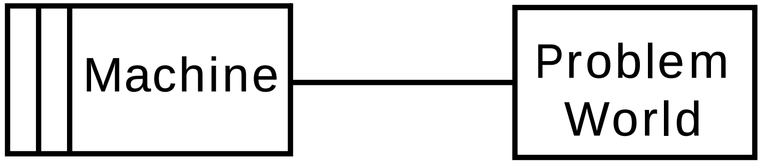
• Control traffic lights	• Lights, roads, traffic, drivers, sensors, …
• Administer a library	• Books, members, fines, catalogue, …
• Web-based retailing	• Goods, delivery company, credit cards, …
• Control a lift	• Doors, sensors, buttons, winding gear, users, …
• Word processing	• Documents, users, …

The Problem World Is Not the Software



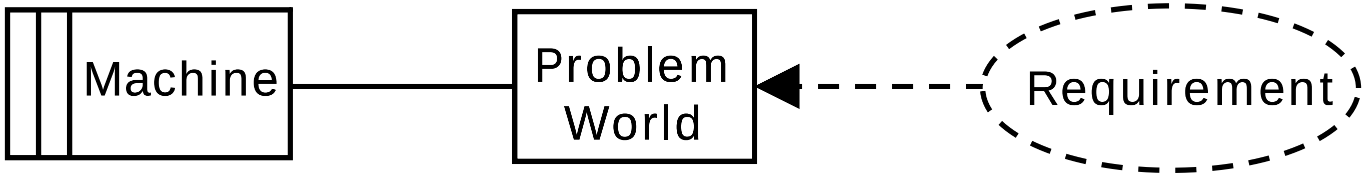
- Software development is building a Machine
  - The developed software running on a computer
- The Machine is connected to the Problem World
  - The Traffic Lights Controller can switch the lights and can monitor the sensors
  - The Library Administration Machine can read bar-coded book cards and magnetic membership cards
  - The Lift Control Machine can turn the motor on, detect button presses, etc

What Kind Of Problem World?



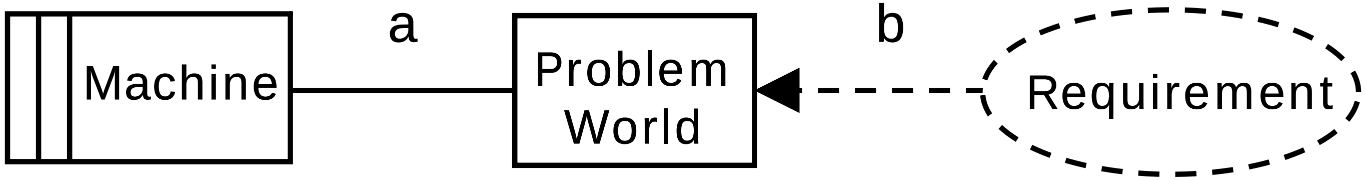
- PFs are (primarily) about problems with a physical problem world
  - Not about factorising prime numbers, finding the spanning tree of a graph, computing the convex hull of a set of points in 3D space, …
- PFs are concerned with physical phenomena
  - A car is in the intersection
  - The book has been borrowed
  - The lift is at floor 2
  - The user has hit the ‘Delete’ key
- Relevant physical phenomena must be designated

The Customer’s Requirement



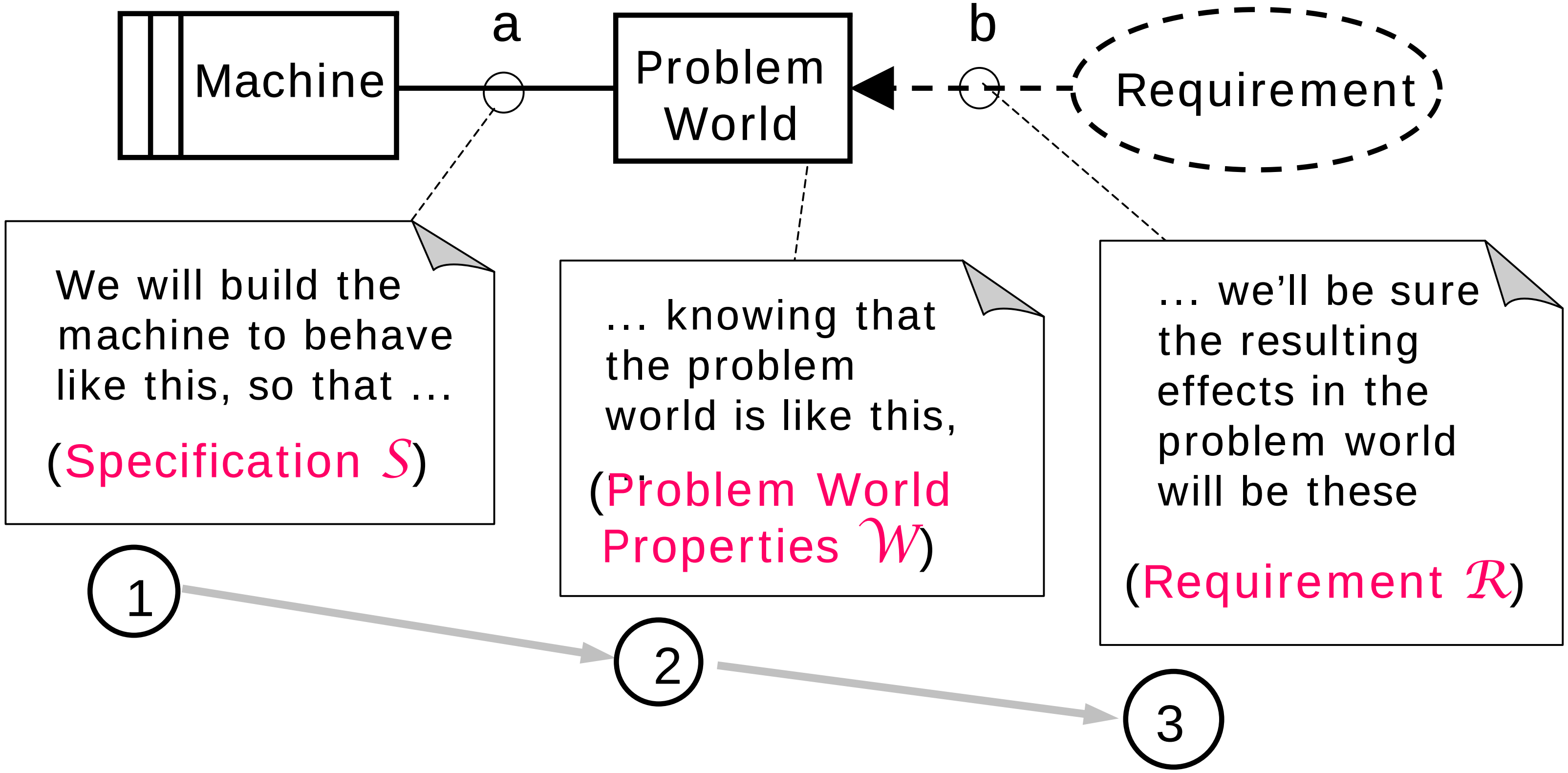
- The customer’s requirement is a condition on the problem world, not on the machine
    - The lift comes when you call it
    - Books are lent only to members
    - Vehicle collisions are prevented
    - When you hit ‘Delete’ the selection (or the character after the cursor) is deleted from the text
    - Confirmed web purchases will be delivered
    - Electricity users are billed only for units used

How Can the Requirement Be Satisfied?



- The machine shares ‘specification’ phenomena a
  - MotorOn, SensorOn, DirnUp, …
- The requirement is about ‘requirement’ phenomena b
  - LiftComes, DoorsOpen, LiftGoesToFloor, …
  - BookIsLent, MembershipExpires, BookIsLost, …
  - VehiclesCollide, VehicleWaits, …
- What connects specification to requirement Phenomena?
  - Problem World properties
    - MotorOn $\wedge$ DirnUp $\Rightarrow$ Lift Rising to Next Floor
    - Book card is fixed to the book

Three Satisfying Descriptions



Machine

a

Problem World

b

Requirement

We will build the machine to behave like this, so that …

(Specification $\mathcal{S}$)

… knowing that the problem world is like this,

(Problem World Properties $\mathcal{W}$)

… we'll be sure the resulting effects in the problem world will be these

(Requirement $\mathcal{R}$)

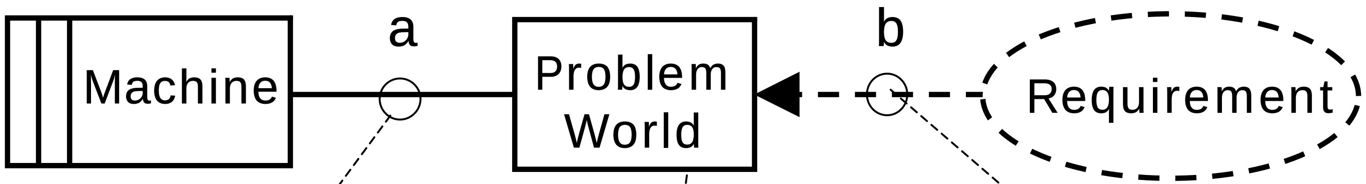
1

2

3

- Three descriptions: $\mathcal{S}, \mathcal{W}, \mathcal{R}$
- The machine specification is adequate iff: $\mathcal{S}, \mathcal{W} \models \mathcal{R}$

Three Satisfying Descriptions: Example Fragment



Machine

a

Problem World

b

Requirement

When button[2] is pressed, and sensor[0] is the most recent sensor on, and MotorOn is false:

set DirnUp and MotorOn; wait until sensor[2] is on; set MotorOff; …

(Machine Specification $\mathcal{S}$)

button[2] is pressed iff a user wants the lift to go to floor 2;

DirnUp and MotorOn causes the lift car to rise;

sensor[2] comes on when the lift is at floor 2;

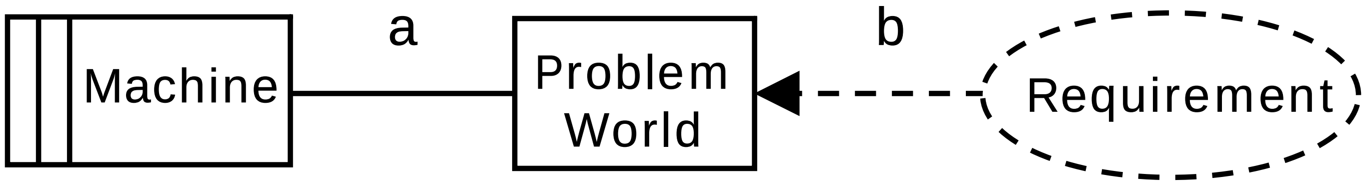
when MotorOn is false lift stops; …

(Problem World Properties $\mathcal{W}$)

When the lift is at one flor, and a user calls the lift to another floor, it comes; …

(Requirement $\mathcal{R}$)

Solving A Problem

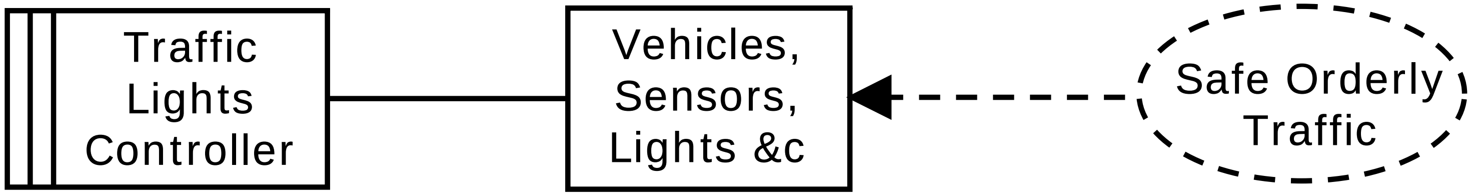


- In the PF approach, ‘solving a problem‘ is devising a machine specification that satisfies the requirement
- We distinguish ‘solving the problem’ from ‘programming’
  - ‘Solving the problem’ gives a machine specification S
  - ‘Programming’ gives a program that satisfies S
- Devising a specification includes problem decomposition
  - Requirement and problem world structures govern the specification (but not necessarily machine) structure
  - The key goal is mastering complexity

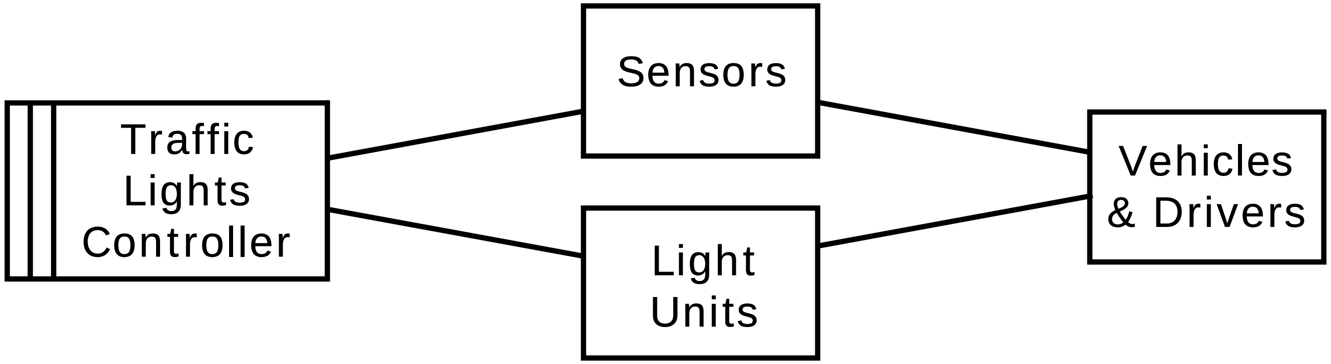
Mastering Complexity

- For a realistic problem
  - The Problem World is complex
  - The Requirement is complex
  - The Machine is complex
- Example: Car central locking
  - Problem World: 4 doors + tailgate, 2 with keys, 4 with buttons + handles, locking control console, ignition on/off, car speed, driver, passengers, parking places, …
  - Requirement: avoid car/contents theft, not locking keys in car, lock doors while running, child-lock setting, doors unlock in crash, shopping convenience, …
  - Machine: ???

Decomposing the Problem World



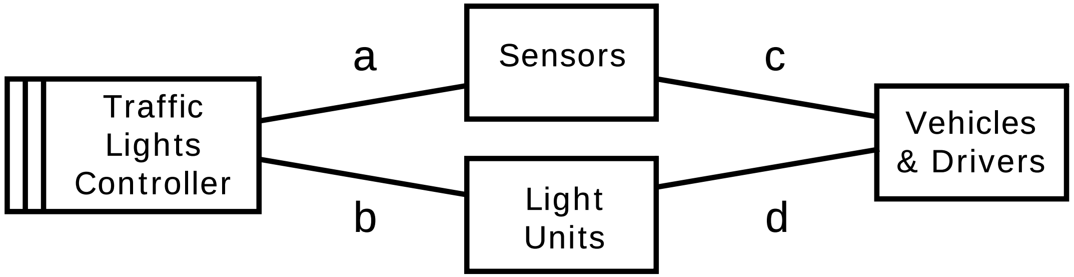
- The Problem World usually demands decomposition



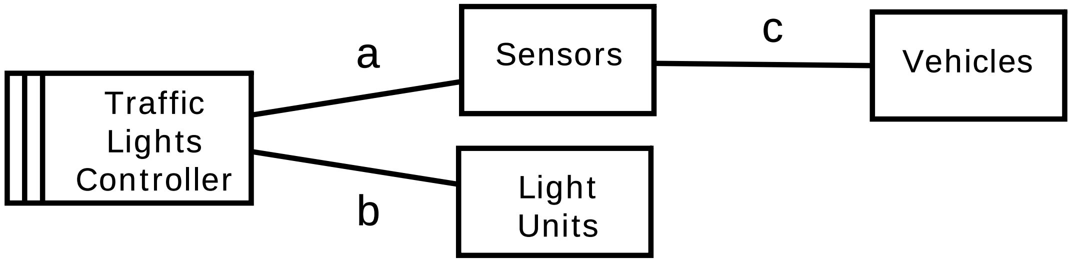
- Domains with interfaces of shared phenomena
  - Allowing structured description of properties
  - Allowing greater clarity of problem scope

Context Diagram: the Problem Scope

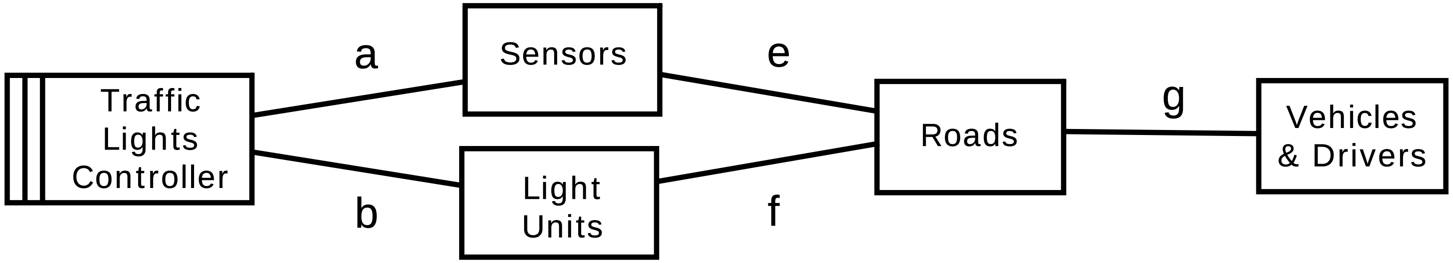
- How is this —



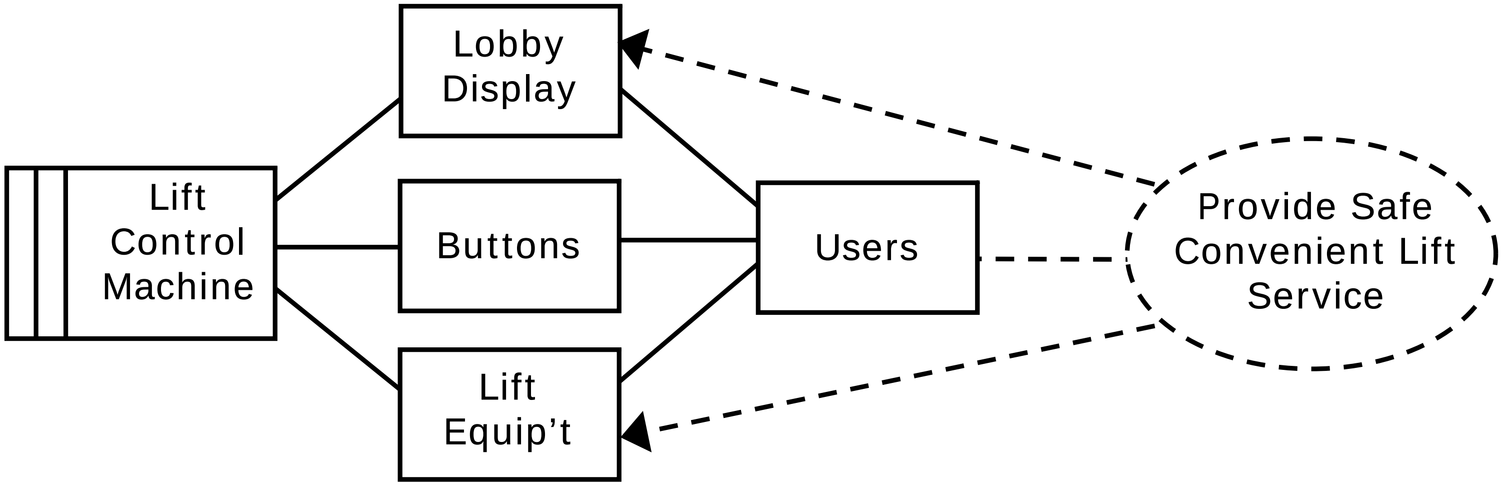
- — different from this?



- — or from this?



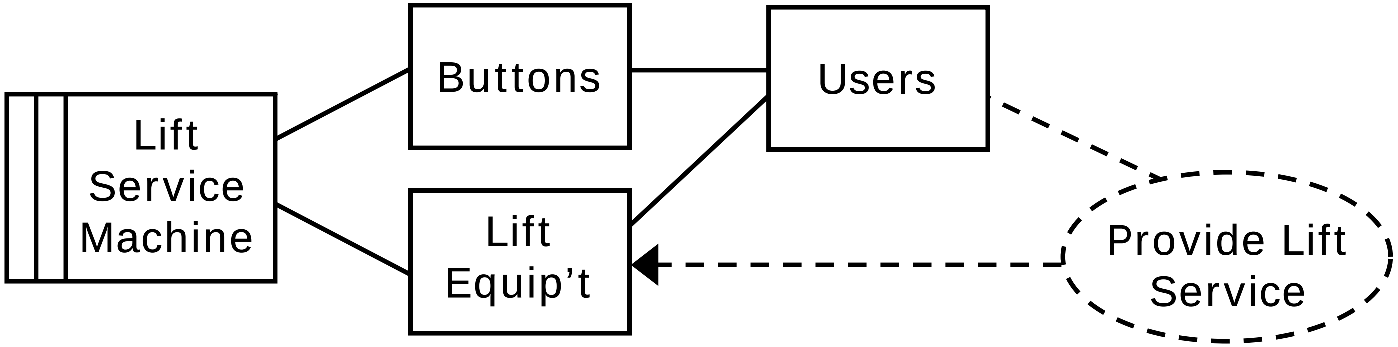
Decomposing the Problem Requirement



- Several 'subrequirements'
  - Provide lift service in response to requests
  - Ensure safety
  - Display lift position on indicator in hotel lobby
- Decompose problem into subproblems
  - Each subproblem has its own Machine, Problem World and Requirement

Problem Decomposition

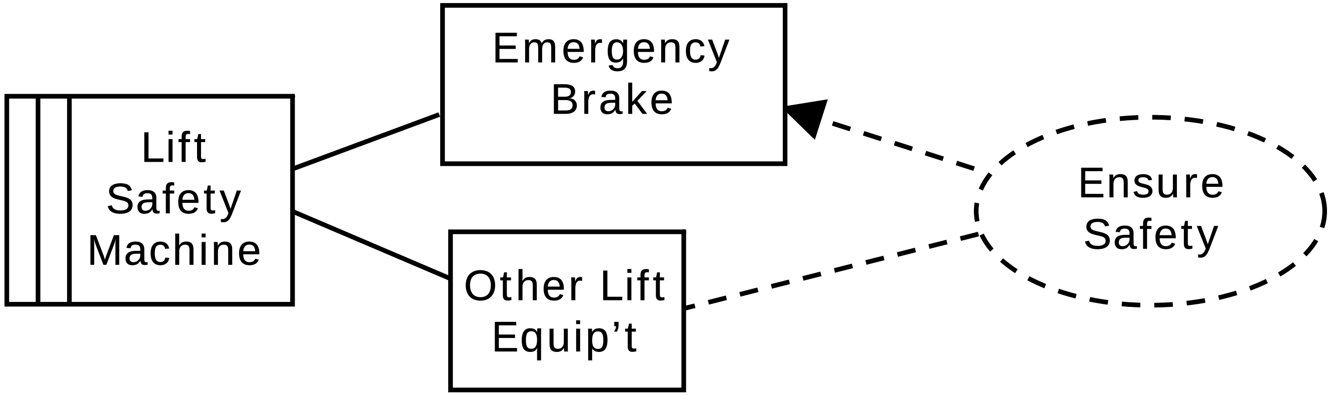
- Subproblem 1: Provide lift service



- The Lobby Display domain is not relevant to the lift service subproblem

Problem Decomposition

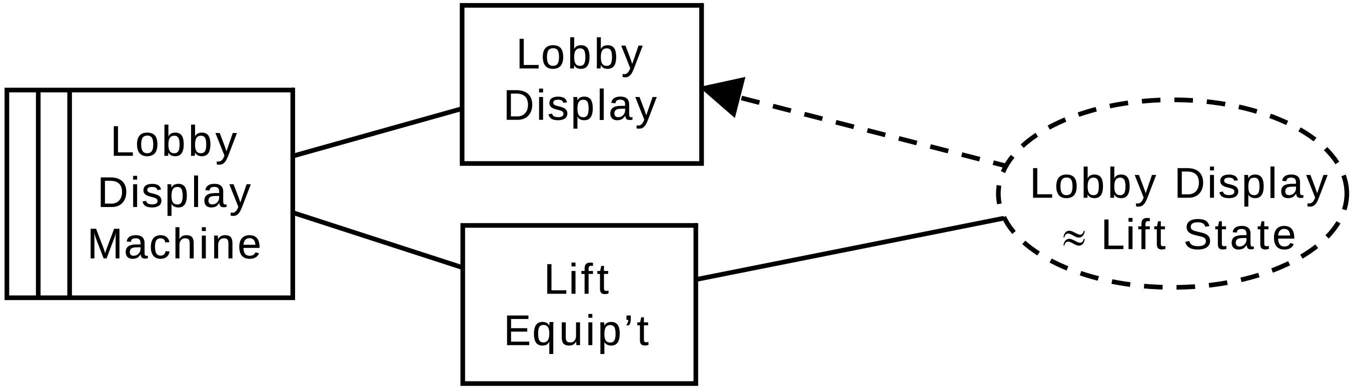
- Subproblem 2: Ensure safety



- The Lobby Display, Users and Buttons domains are not relevant to the lift safety subproblem
- The Lift Equipment domain must be further decomposed for the lift safety subproblem

Problem Decomposition

- Subproblem 3: Maintain lobby display

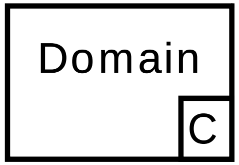


- The Users and Buttons domains are not relevant to the maintain lobby display subproblem

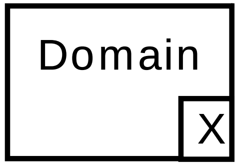
Problem Decomposition Principles

- Decomposition does not add to the Problem World
  - Subproblem Machine, Problem World and Requirement are contained in the whole
  - Subproblems may view problem domains differently
    - Lift Service controls Lift Equipment
    - Lobby Display monitors Lift Equipment
- Decomposition is guided
  - Ideally the subproblems fit known Problem Frames
    - Standard (intuitive) class of requirement
    - Standard minimal Problem World decomposition
    - Standard characteristics of problem domains
    - Standard control patterns for phenomena
    - Standard decompositions into sub-sub-problems
    - Standard subproblem concerns
  - There are also important decomposition heuristics

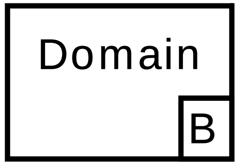
Domain Characteristics



- Causal:
  - Causal phenomena (C) eg events, states
  - Internal causality relationships



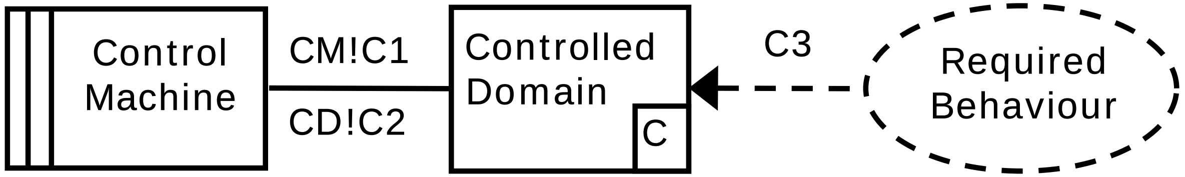
- Lexical:
  - Symbolic (Y) phenomena eg ints, chars
  - Reification gives a causal infrastructure



- Biddable:
  - Human, causal phenomena
  - No fully reliable internal causality
  - Can be ‘bidden’ to follow a procedure

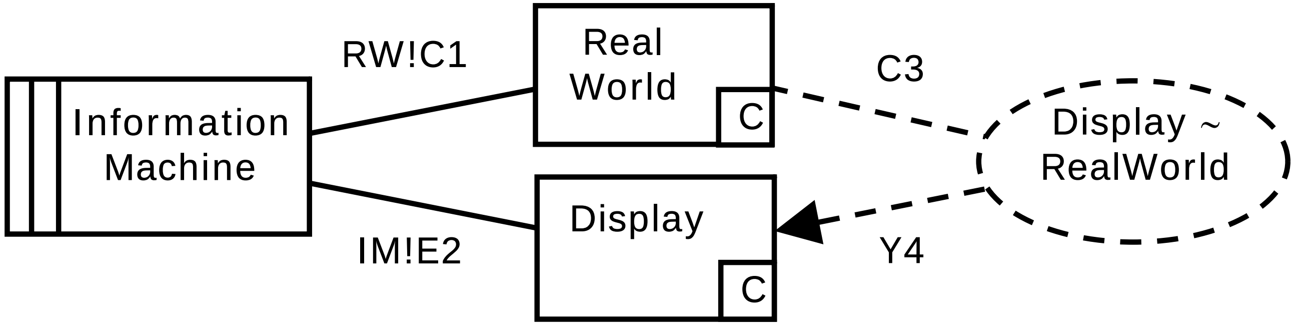
- Domain characteristics matter: they affect concerns
  - eg: lexical domain has no reliability concern

Problem Frame: Required Behaviour



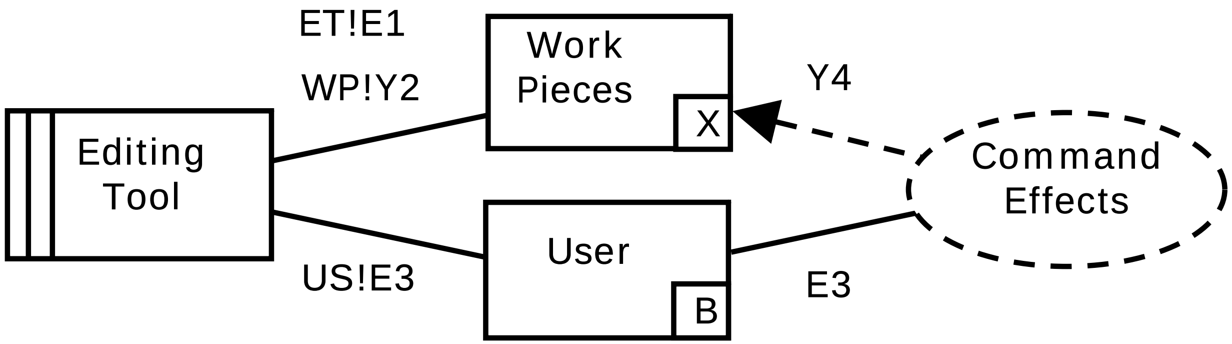
- The intuition:
  - Achieve/maintain a required behaviour `in a given problem domain
- The problem parts:
  - Controlled Domain: causal domain
  - Interface: C1, C2 are causal phenomena
  - Required Behaviour: behaviour of CD wrt C3
- Decomposition shown is minimal
  - Controlled Domain can be further decomposed

Problem Frame: Information Display



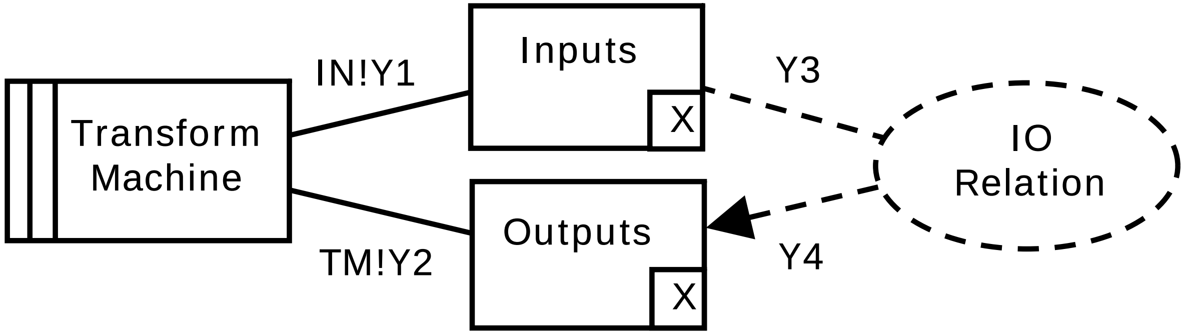
- The intuition:
  - Display information about a part of the world
- The problem parts:
  - Real World: autonomous, active, causal
  - Display: state-reactive, symbolic display
  - Display ~ RW: Display Y4 to correspond to RW C3
- Decomposition shown is minimal
  - Real World and Display can be further decomposed

Problem Frame: Simple Workpieces



- The intuition:
  - Provide a tool for editing texts, graphics, etc
- The problem parts:
  - Workpieces: event-reactive, inert, lexical
  - User: autonomously active
  - Command Effects: effects in WP of User commands
- Decomposition shown is minimal
  - No further decomposition

Problem Frame: Transformation

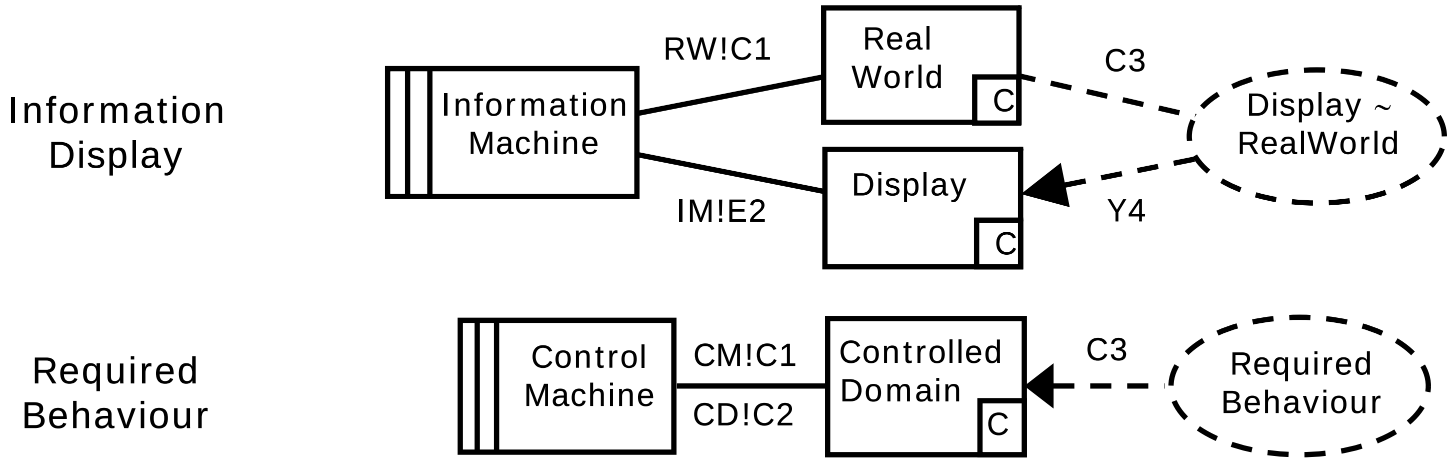


- The intuition:
  - Transform input data to output data
- The problem parts:
  - Inputs: lexical
  - Outputs: lexical
  - IO Relation: correspondence of Outputs to Inputs
- Decomposition shown is minimal
  - Inputs and Outputs can have >1 source and sink

Standard Frame Decompositions

- In the simplest form of a frame
  - The machine is a one-module program
  - The frame concern and other particular concerns demand care in the specification of the one module
- In a more complex form
  - Some concern demands further decomposition into sub-sub-problems
  - Standard decompositions (perhaps for >1 frame)
  - Composition of the resulting machines is standard
- In the most complex form
  - The frame captures a class of problems that are
    - Relatively large and complex, but …
    - … now very well understood and standardised
  - Examples: 1970s compiler, MVC, …

A Standard Decomposition



Information Display

Information Machine

RW!C1

IM!E2

Real World

C

Display

C

C3

Y4

Display ~ RealWorld

Required Behaviour

Control Machine

CM!C1

CD!C2

Controlled Domain

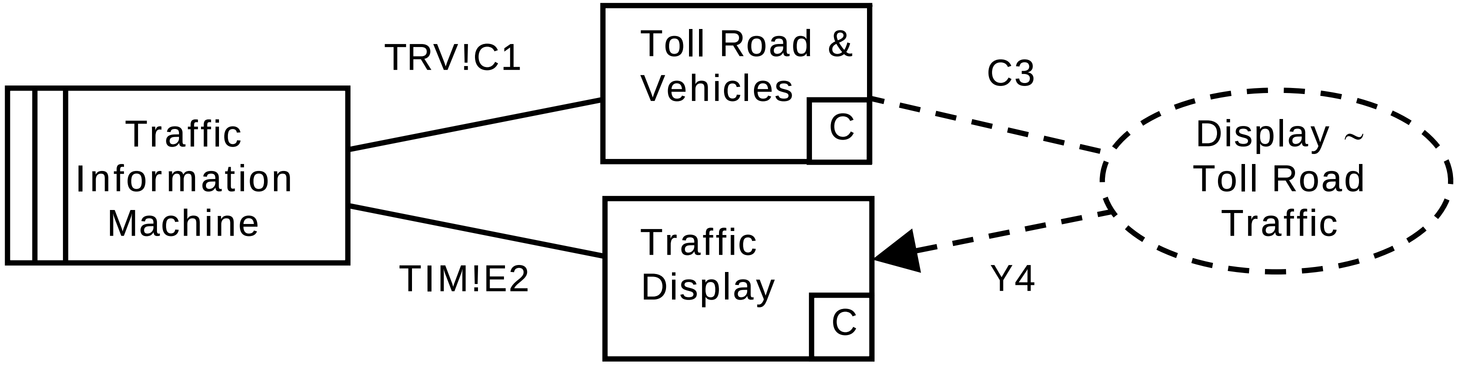
C

C3

Required Behaviour

- In both frames the Machine gathers information by C1
- The C1 information may be untimely
  - Needing to be stored, processed and accumulated
- The decomposition introduces a model domain
  - One sub-sub-problem to build and maintain model
  - One sub-sub-problem to use information from model

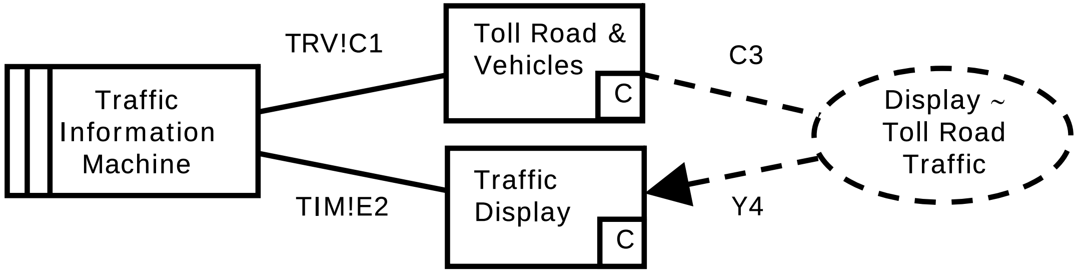
Displaying Toll-Road Usage



- Vehicles' badges are read (C1) at entry and exit points
- The Traffic Display must show (Y4):
  - Number of vehicles currently on road (C3)
  - Shortest and longest traversal times (C3)
  - … (C3)
- Information in C1 events must be stored and processed for use in computing updates (E2) to Traffic Display
- We must introduce a Traffic Model domain
  - It is essentially a local variable of the Machine

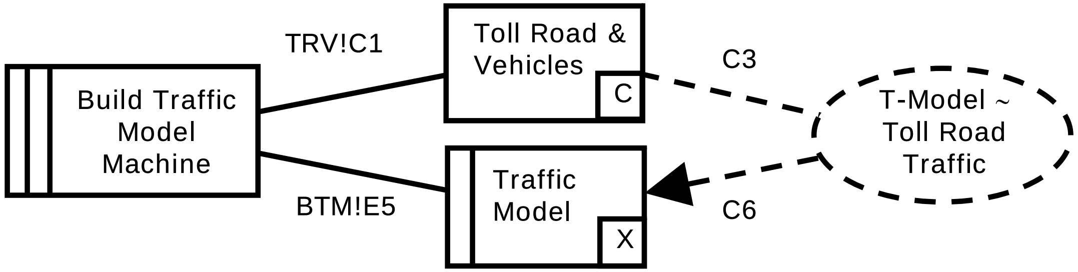
Displaying Toll-Road Usage

Original Problem

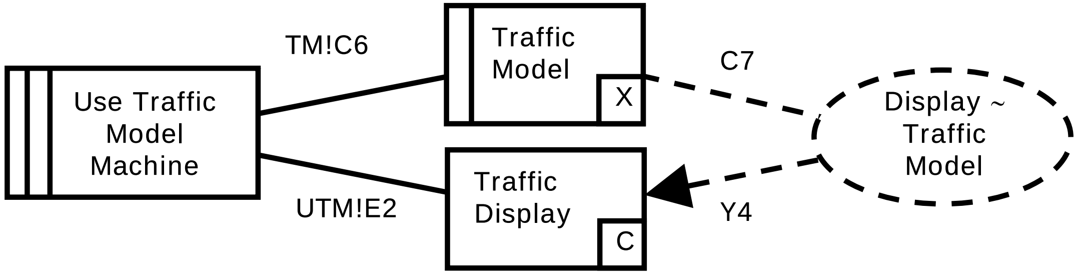


Decomposed

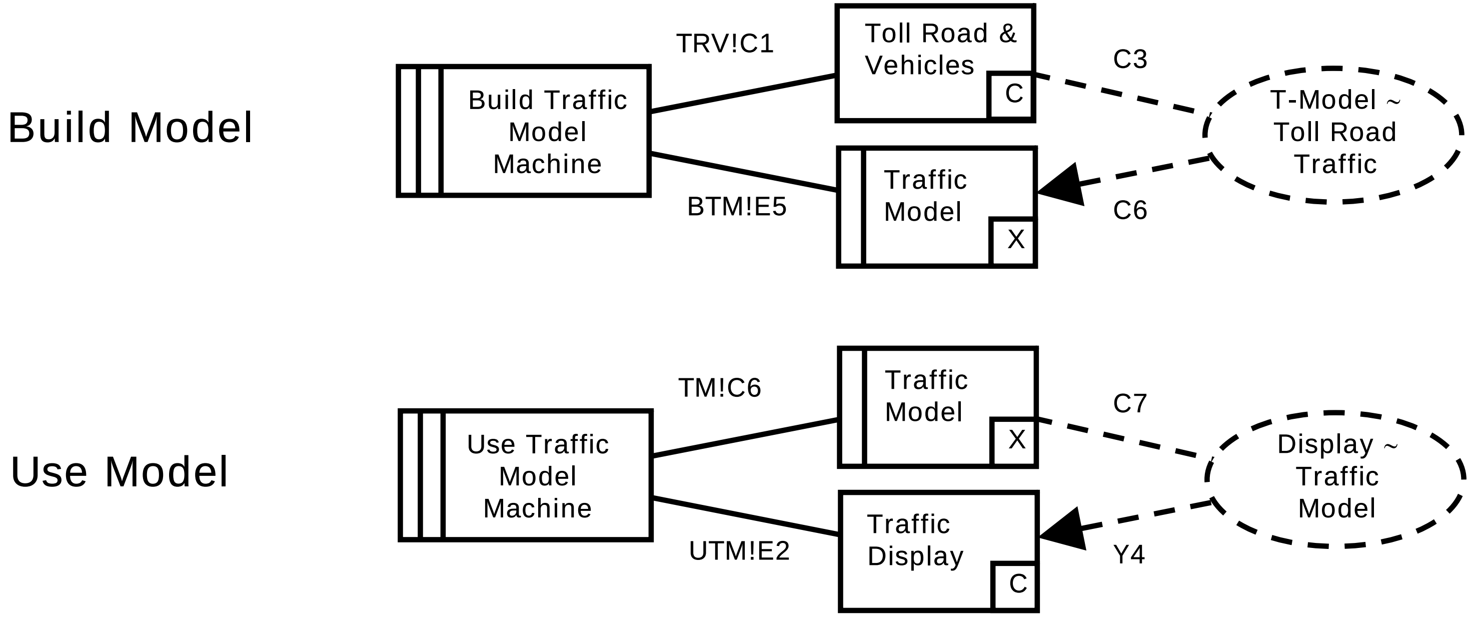
Build Model



Use Model



Displaying Toll-Road Usage



Build Model

Build Traffic Model Machine

TRV!C1

Toll Road & Vehicles

C

BTM!E5

Traffic Model

X

C3

C6

T-Model ~ Toll Road Traffic

Use Model

Use Traffic Model Machine

TM!C6

Traffic Model

X

UTM!E2

Traffic Display

C

C7

Y4

Display ~ Traffic Model

- Traffic Model is part of Machine in original problem, part of Problem World in decomposed subproblems
- Design of model: What questions must it answer?
  - For large persistent models (eg databases) the model defines the envelope of all questions that can be answered

A Tutorial on

Software Development

Problem Frames

Part 2

BCS RESG Problem Frames Day
Open University
10 May 2006

Michael Jackson
jacksonma@acm.org

A Tutorial In Two Parts

10.00-11.15 Part 1

Software development problems

Where is the problem?

Solving a problem

Problems and subproblems

11.30-12. 45 Part 2

Subproblem concerns

Subproblem composition

Composition concerns

Normal and radical design

Discussion

Additional Slides: Problem Frames Bibliography

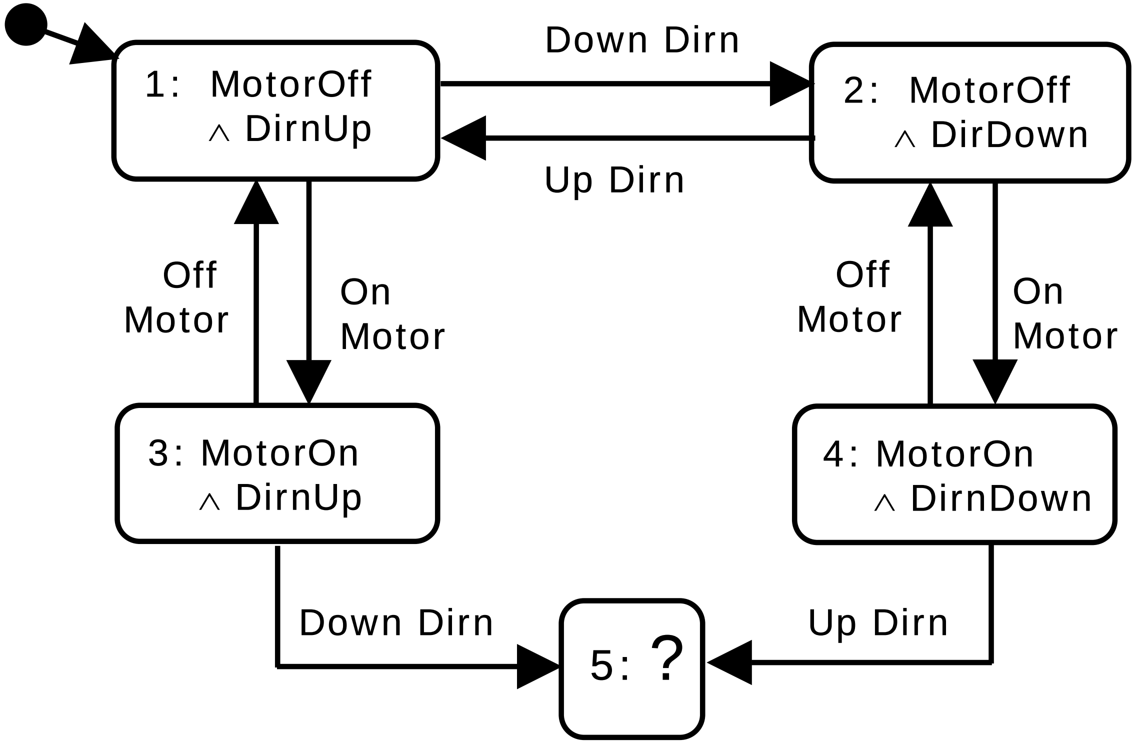
Concerns

- A concern is any matter to which the developers must pay attention to obtain a good solution
  - The basic ('frame') concern is $\mathcal{S}, \mathcal{D} \models \mathcal{R}$
  - There are also other more specific concerns that must be addressed to avoid serious failures
- Concerns are mostly about things going wrong
  - Engineers pay a lot of attention to failures
  - A vital part of engineering know-how:
    - Which concerns are important?
    - How to address them effectively?
- A rough distinction
  - Subproblem concerns
  - Composition concerns

Some Standard Subproblem Concerns

- Breakage
  - The machine breaks a problem domain
- Initialisation
  - Incompatible initial states of machine and world
- Identities
  - Interacting with the wrong member of a set
- Reliability
  - Problem domain properties are not satisfied
- Completeness
  - Some Problem World conditions are ignored

Breakage Concern in Lift Control



- Dirn must not be switched while the motor is on

Initialisation Concern

- Program initialisation: a known concern
  - Avoiding uninitialised variables, array indices, &c
- System initialisation: less known, much harder
  - When Machine execution begins, is the Problem World in a compatible state?
- Examples:
  - Lift Control
  - Toll-Road Traffic Display
  - Library Administration
  - Bank Accounts
  - Car Park Control

Initialisation Techniques

- Method 1
  - Machine initial behaviour is correct in any world state
- Method 2
  - Subproblem Machine forces correct initial world state
- Method 3
  - Machine detects world state, self-initialises to fit
- Method 4
  - Operator sets correct world state before switch-on
- Method 5
  - Operator detects world state, initialises machine to fit
- Method 6
  - Operator sets correct world state incrementally
- Method 7
  - World converges automatically to Machine state

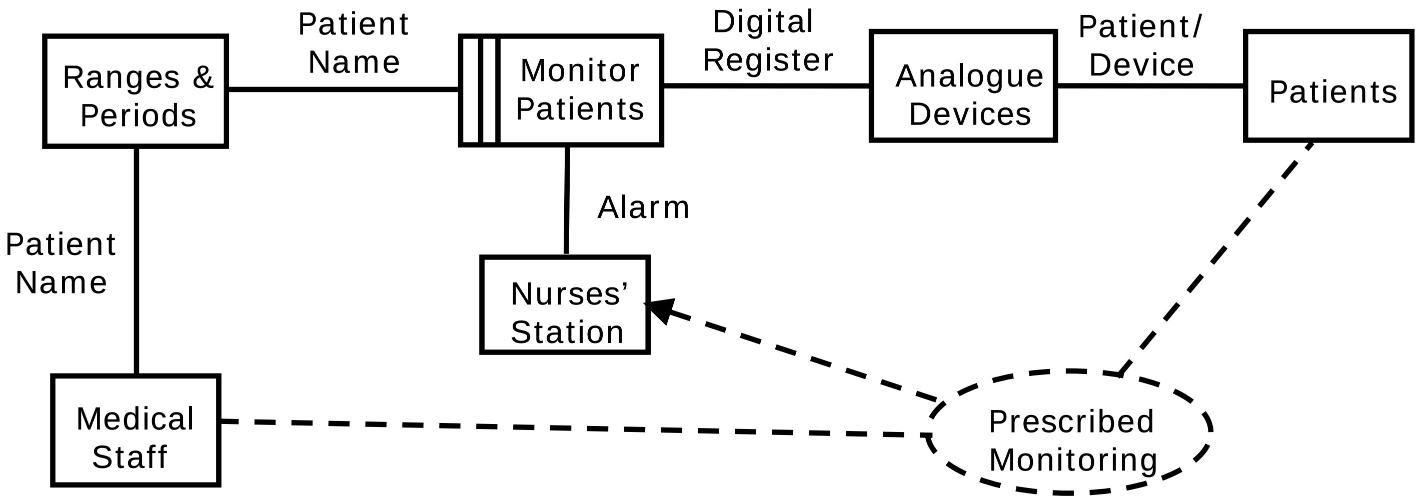
Identities Concern

- A multiple domain contains a set of individuals
  - eg: Bank accounts, lift floors, ICU patients
- When the machine interacts with individuals of the set it's necessary to ensure it's the right individual
  - Monitoring the wrong ICU patient is catastrophic

It turned out that two wires connecting the CAP's sidestick to one Elevator and Aileron Computer (ELAC), of which there are two, had been reverse-connected during maintenance, and the fault had been discovered neither by post-maintenance check, nor by post-maintenance cross-check, nor by the flight crew's pre-take-off control system check.

Peter B Ladkin
A320 Incident
http://catless.ncl.ac.uk/Risks/23.24.html#subj12.1

Identities Concern: An Example

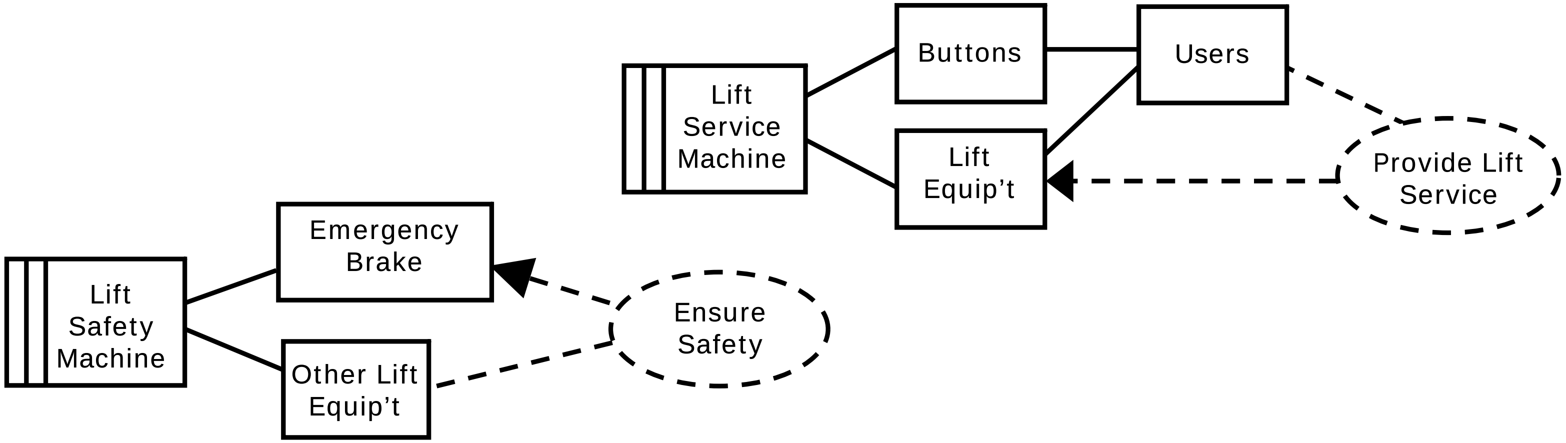


- How to identify monitored patient reliably?
  - Patients have names, and are in beds in ICU
  - Medical staff use names in prescribing monitoring
  - Devices (eg thermometers) are attached to patients
  - Devices are connected to machine's registers
- What is static, what is dynamic here?

Reliability Concern

- Problem domains are parts of the physical world
- Domain descriptions capture their properties
- We rely on these properties in devising the machine …
  … so the system relies on them to function correctly
- But nothing in the physical world is perfectly reliable
  - Design and build are only approximate
    - The world is not a formal system
  - Time takes its toll of decay
    - Springs weaken
    - Metal rusts, concrete weathers
    - Gears and bearings wear out
    - …

Reliability Concern In Lift Control



- Lift Service Machine assumes 'healthy' equipment
- Lift Safety Machine:
  - Monitors equipment for 'healthiness'
  - Applies Emergency Brake if a fault is detected
- The Lift Service and Lift Safety Machines assume different Problem World properties

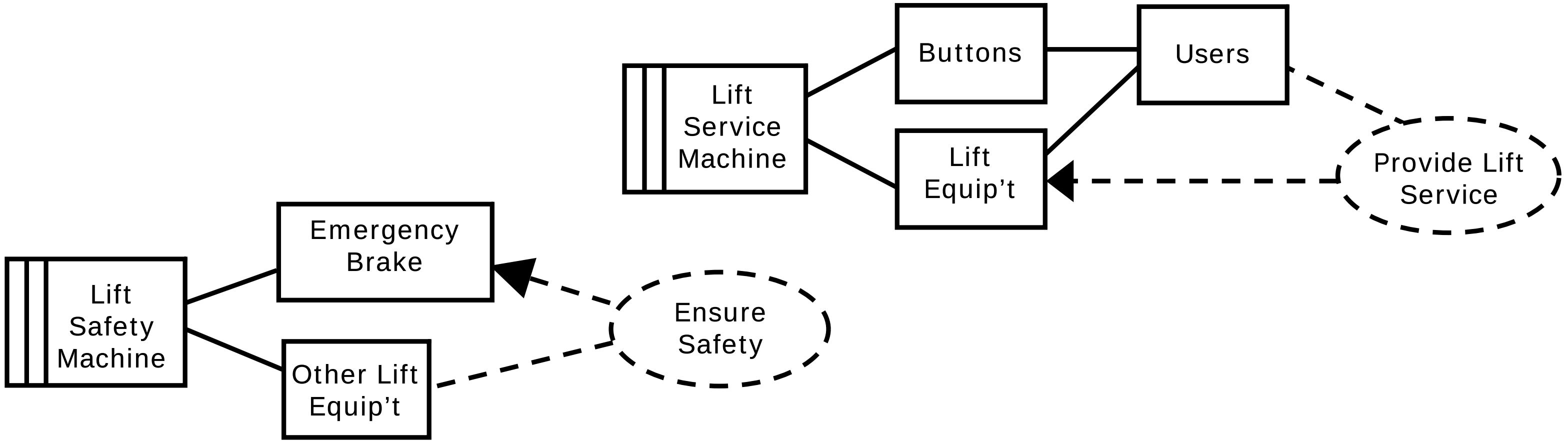
Completeness Concern

- Physical (and especially human) Problem Worlds
  - Unbounded possibilities of state and behaviour
  - It’s hard to take enough possibilities into account
- In Library Administration
  - A Member dies
  - A book is stolen from the library shelves
- In Toll-Road Traffic Display
  - Broken down Vehicle lifted on to a breakdown truck
  - Car in crash falls off toll-road at a bridge
- In Patient Monitoring
  - Two child patients exchange monitoring devices
  - Patient marries and changes name
- In Traffic Light Control
  - Vehicle breaks down with wheel on sensor
  - Naughty children jump up and down on sensors

Subproblem Composition

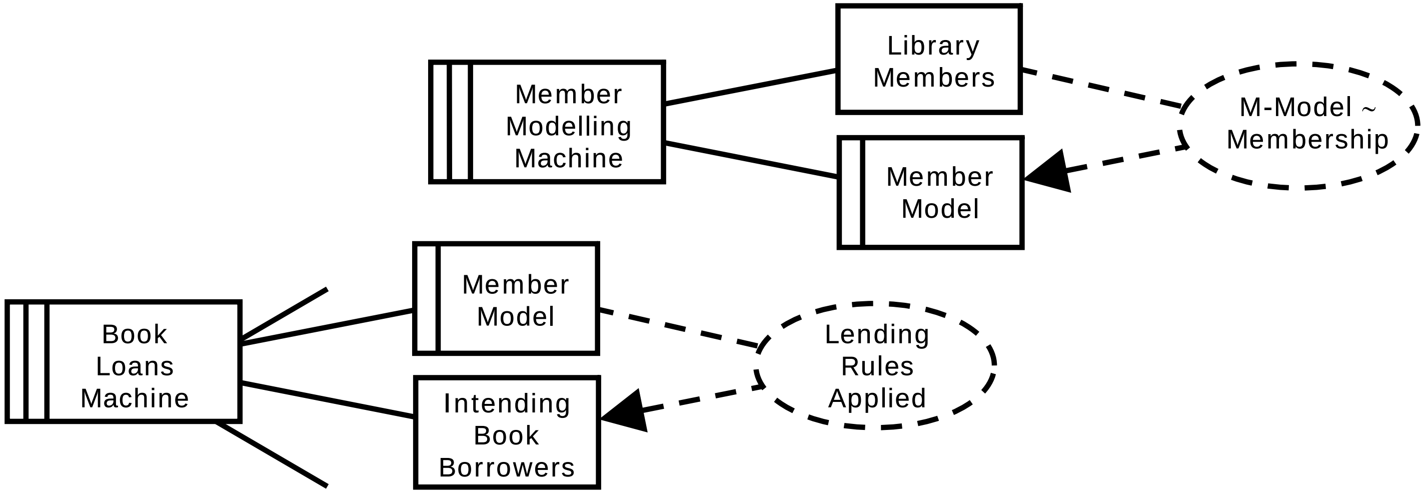
- Decomposition is only half the job
  - Decomposed parts must be composed into a whole
- PF decomposition does not assume
  - Standard form of subproblem (eg object)
  - Standard form of connection (eg method invocation)
- Composition means any and all of:
  - Composing subproblem requirements
  - Composing subproblem domains
  - Composing subproblem phenomena control
  - Composing subproblem machines
- Subproblem composition is an explicit set of tasks
  - With its own composition concerns
- Composition may require changing a subproblem

Composing Subproblem Requirements



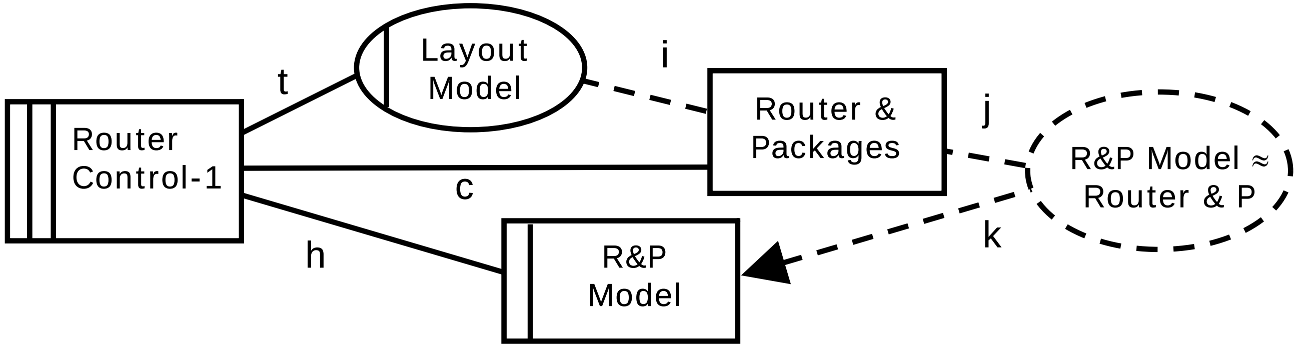
- Lift Service requirement
  - When button is pressed, lift comes
- Lift Safety requirement
  - When equipment is faulty, halt and apply brake
- These requirements will sometimes conflict
  - Customer must decide precedence

Composing Subproblem Requirements



- Dynamic Member Modelling requirement
  - Build dynamic model of fees, memberships, etc
- Book Lending requirement
  - Use static member model, lend only to members
- The Book Lending requirement needs refinement
  - 2-week loan if membership expires in 1 week?
  - Loan renewal after membership expires?

Composing Subproblem Domains

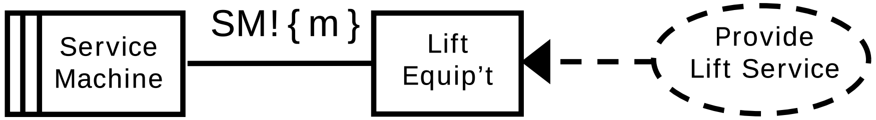


- 2 Package Router models answering different questions
  - Layout: topology of pipes, switches, sensors, bins
  - R&P: packages queueing in pipes and switches
- These model domains can be combined
  - Static aspects: topology
  - Dynamic aspects: package movement

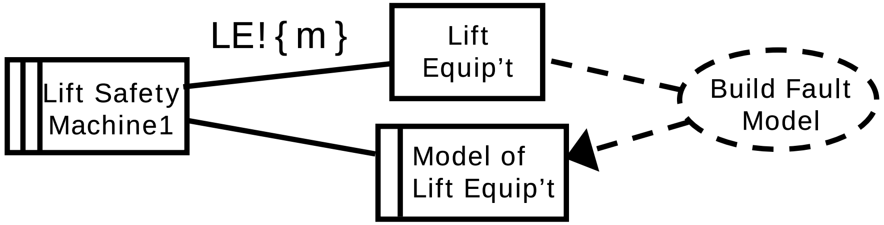
Composing Subproblem Phenomena Control

Subproblems view Lift Equip't MotorOn control differently

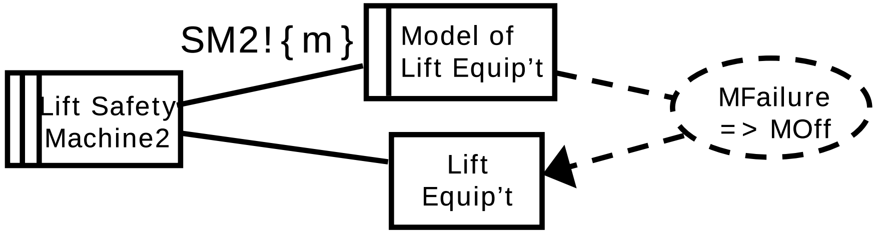
- Lift Service Machine controls MotorOn



- For Safety Machine 1 Lift Equipment controls MotorOn



- Safety Machine 2 controls MotorOn



Composing Subproblem Machines

- Subproblem machine execution relationships
  - Concurrency
    - Model-building concurrent with model use
    - Lift position display concurrent with lift service
    - Library membership concurrent with book loans
  - Sequentiality
    - Initialise lift equipment; provide lift service
    - Avionics: taxi; take-off; climb; cruise; …
  - Choice
    - Provide lift service
      □ maintain safe stationary state
    - Provide toll-road information
      □ update road/booth mappings

Composing Subproblem Machines

- Architecture frames (Hall, Rapanotti et al)
  - eg MVC pattern: workpiece with manipulable display
- Composition controllers (Laney et al)
  - Interposed between machine and problem domains
  - eg: arbitrating between lift service and lift safety
- Concurrency constrained by designed domains
  - eg: mutual exclusion between model build and use
- Program transformation technologies
  - Merging machines into combined sequential process

    eg: combine lift service with build fault model
  - Moving locus of control

    eg: implement reader as subroutine of writer

Some Composition Concerns

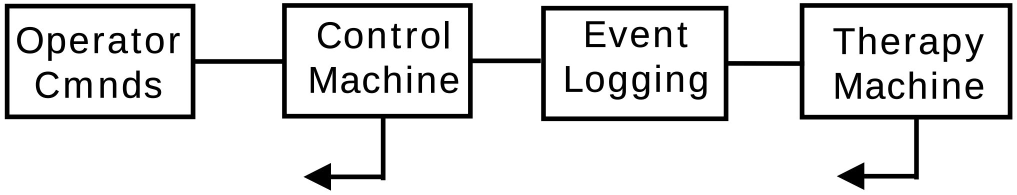
- Interference
- Interleaving
- Requirement conflict
- Switching
- Inconsistent problem world descriptions
- Different abstraction granularities
- Redundancy (eg same model twice)
- Sharing (eg a screen)
- Criticality ordering

Switching

- A typical switching concern
  - Switching from normal operation to fault handling
    - When can the normal operation machine relinquish control of the domain?
    - When can the fault-handling machine take over control of the domain?
- Relinquishing control
  - The domain must be left in a ‘safe‘ state
    - Juggler can’t stop while a ball is in the air
  - A termination concern for relinquishing machine
- Taking over control
  - The initialisation concern must be addressed for the machine taking over control

Criticality Ordering

- What is a dependable system?
  - Every requirement is dependable?
  - More critical requirements are more dependable?
- Composition must respect criticality ordering
  - Of requirements
  - Of machine design and implementation
- An implementation example:



- All events caused at Therapy Machine are to be logged
  - So Event Logging machine ‘sees’ all these events
  - But …

Deferring Composition Concerns

- Why defer composition concerns?
  - Because composition concerns need explicit and separate consideration and treatment
  - Because familiar subproblem classes complicated by composition concerns are harder to recognise
  - Because familiar subproblem classes complicated by composition concerns are harder to solve
  - Because composition is hard to deal with if you don’t know what you are composing
  - Because some compositions are hard to deal with if you have to consider each component separately
- Premature composition causes needless complexity
  - Deferring composition exposes minimal complexity

Analysis and Implementation

- There are strong traditions of ‘seamless’ development
  - Formal approaches based on refinement
  - Informal approaches based on ‘modelling’
- Does the PF approach support ‘seamless’ development?
  - Subproblem Machines are components to be assembled (after addressing composition concerns)
  - What assembly techniques do we have?
  - What assembly techniques can we develop?
- What if we can’t do ‘seamless’ development?
  - PF analysis establishes properties of projections of completed system (Machine + Problem World)
  - cf: Program slicing?

Encouraging Normal Design

- Normal design
  "… the engineer knows at the outset how the device in question works, what are its customary features, and that, if properly designed along such lines, it has a good likelihood of accomplishing the desired task"
- Operational principle (Polanyi) of the device
  "How its characteristic parts … fulfil their special function in combining to an overall operation which achieves the purpose"
- Normal configuration
  "… the general shape and arrangements [of parts] that are commonly agreed to best embody the operational principle"

W G Vincenti: What Engineers Know and How They Know It
Johns Hopkins University Press paperback edn 1993

Thank you

Problem Frames Bibliography (Last update: 28/12/2005)

- [Armstrong 02] Steve Armstrong and Kathy Maitland; Reconditioning Use Cases with Problem Frames; (unpublished).
- [Barroca 03] Leonor Barroca, J L Fiadeiro, M Jackson, R Laney and B Nuseibeh; Problem Frames: A Case for Coordination; in Proceedings of Coordination 2004: 6th International Conference on Coordination Models and Languages, Pisa, 2004.
- [Barroca 04a] Leonor Barroca, J L Fiadeiro, M Jackson and R Laney; Dynamic Assembly of Problem Frames; submitted to (but not accepted by) the ACM SIGSOFT conference on the Foundations of Software Engineering (FSE04), 2004.
- [Bjorner 97] D Bjorner, S Koussoube, R Noussi, G Satchok; Michael Jackson's problem frames: towards methodological principles of selecting and applying formal software development techniques and tools; in Proceedings of 1st International Conference on Formal Engineering Methods (ICFEM '97), November 12-14, 1997, Hiroshima, JAPAN.
- [Bleistein 04] Steven J Bleistein, Karl Cox and June Verner; Problem Frames Approach for e-Business Systems; in Proceedings of the 1st International Workshop on Advances and Applications of Problem Frames, 2004.
- [Bleistein 05] Steven J Bleistein, Karl Cox and June Verner; Validating strategic alignment of organizational IT requirements using goal modeling and problem diagrams; Journal of Systems and Software, to appear.
- [Bray 03] Ian K Bray and Karl Cox; The Simulator; Another, Elementary Problem Frame? in Proceedings of the 9th International Workshop on Requirements Engineering: Foundation For Software Quality (REFSQ03), 2003.
- [Bray 04] Ian K Bray; Experiences of Teaching Problem Frame Based Requirements Engineering to Undergraduates; in Proceedings of the 1st International Workshop on Advances and Applications of Problem Frames, 2004.
- [Brier 04] John Brier, Lucia Rapanotti and Jon G Hall; Problem Frames for Socio-technical Systems: predictability and change; in Proceedings of the 1st International Workshop on Advances and Applications of Problem Frames, 2004.
- [Choppy 04] Christine Choppy and Gianna Reggio; A UML-Based Method for the Commanded Behaviour Frame; in Proceedings of the 1st International Workshop on Advances and Applications of Problem Frames, 2004.

- [Cox 03] Karl Cox and Keith Phalp; From Process Model to Problem Frame: A Position Paper; in Proceedings of the 9th International Workshop on Requirements Engineering: Foundation For Software Quality (REFSQ03), 2003.
- [Cox 04] Karl Cox, Keith Phalp, Aybüke Aurum, Steve Bleistein and June Verner; Connecting Role Activity Diagrams to the Problem Frames Approach; AWRE'04 9th Australian Workshop on Requirements Engineering, 2004.
- [Cox 05] Karl Cox, Jon G Hall and Lucia Rapanotti; A Roadmap of Problem Frames Research; Information and Software Technology 2005, to appear.
- [Delannay 02] G Delannay; A Meta-model of Jackson's problem frames; Technical Report, University of Namur, 2002.
- [Gunter 99] Carl A Gunter, Elsa L Gunter, Michael Jackson and Pamela Zave; A Reference Model for Requirements and Specifications; Proceedings of ICRE 2000, Chicago Ill, USA; re-printed in IEEE Software Volume 17 Number 3, pages 37-43, May/June 2000.
- [Hall 02] Jon G Hall, Michael Jackson, Robin Laney, Bashar Nuseibeh and Lucia Rapanotti; Relating Software Requirements and Architectures Using Problem Frames; in Proceedings of the 2002 International Conference on Requirements Engineering (RE'02), Essen, 2002.
- [Hall 03a] Jon G Hall and Andrés Silva; A Requirements-based Framework for the Analysis of Socio-technical System Behaviour; in Proceedings of Ninth International Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'03); Essener Informatik Beiträge, pages 117-120, 2003.
- [Hall 03b] Jon G Hall and Lucia Rapanotti; A Reference Model for Requirements Engineering; in Proceedings of the 11th Joint International Conference of Requirements Engineering (RE'03), 2003.
- [Hall 04a] Jon G. Hall, Lucia Rapanotti and Michael Jackson; Problem frame semantics for software development; Software and Systems Modeling, Springer Volume 4 Number 2, pages 189-198, May 2005.
- [Hall 04b] Jon G Hall and Lucia Rapanotti; Problem Frames for Socio-Technical Systems; in Requirements Engineering for Socio-Technical Systems, eds Andrés Silva and José L Maté; Idea Publishing Group, 2004.

- [Hall 05a] Jon G Hall, Lucia Rapanotti, Karl Cox, Steven Bleistein and June Verner; Problem Frames Analysis of a Complex Organisational Problem; Proceedings of RE05, The International Requirements Engineering Conference, Paris 2005.
- [HayesJJ 03] Ian J Hayes, Michael A Jackson and Cliff B Jones; Determining the specification of a control system from that of its environment; in Keijiro Araki, Stefani Gnesi and Dino Mandrioli eds, Formal Methods: Proceedings of FME2003, pages 154-169, Springer Verlag, Lecture Notes in Computer Science 2805, 2003.
- [JacksonDM 96] Daniel Jackson and Michael Jackson; Problem Decomposition for Reuse; Software Engineering Journal 11,1 19-30, January 1996.
- [JacksonM 94c] Michael Jackson; Problems, Methods and Specialisation; SE Journal Volume 9 Number 6 pages 249-255, November 1994; edited and abridged in IEEE Software Volume 11 Number 6 pages 57-62, November 1984.
- [JacksonM 95a] Michael Jackson; Problems and Requirements; a Keynote Address at RE'95; in Proc RE'95, pages 2-8; IEEE CS Press, 1995.
- [JacksonM 95b] Michael Jackson; The World and the Machine; a Keynote Address at the 17th International Conference On Software Engineering; in Proceedings of the 17th Interna-tional Conference On Software Engineering, p283-292, ACM and IEEE CS Press, 1995.
- [JacksonM 95c] Michael Jackson; Problem Architectures; a Position paper for the International Conference On Software Engineering Workshop on Architectures for Software Systems; in Cooperation with the 17th International Conference On Software Engineering, pages 138-147, 1995.
- [JacksonM 95d] Michael Jackson and Pamela Zave; Deriving Specifications from Requirements: An Example; in Proceedings of the 17th International Conference On Software Engineering, pages 15-24, ACM and IEEE CS Press, 1995.
- [JacksonM 95e] Michael Jackson; Software Requirements & Specifications: A Lexicon of Practice, Principles, and Prejudices; Addison-Wesley, 1995.
- [JacksonM 99a] Michael Jackson; Problem Analysis Using Small Problem Frames; Proceedings of WOFACS ’98, Special Issue of the South African Computer Journal Volume 22, pages 47-60, March 1999.

- [JacksonM 99f] Michael Jackson; The Real World; in Millennial Perspectives in Computer Science; Jim Davies, Bill Roscoe, Jim Woodcock eds; Proceedings of the 1999 Oxford-Microsoft symposium in honour of Sir Antony Hoare, pages 157-173; Palgrave, Basingstoke, England, 2000.
- [JacksonM 00b] Michael Jackson; Problem Analysis and Structure; in Engineering Theories of Software Construction, Tony Hoare, Manfred Broy and Ralf Steinbruggen eds; Proceedings of NATO Summer School, Marktoberdorf; IOS Press, Amsterdam, Netherlands, August 2000.
- [JacksonM 00c] Michael Jackson; Aspects of System Description; in Programming Methodology, A McIver and C Morgan eds; Springer Verlag, 2003.
- [JacksonM 00d] Michael Jackson; Problem Frames: Analysing and Structuring Software Development Problems; Addison-Wesley, 2000.
- [Jacksonm 02a] Michael Jackson; Some Basic Tenets of Description; Software & Systems Journal Volume 1, Number 1, pages 5-9, September 2002.
- [JacksonM 03a] Michael Jackson; Why Program Writing Is Difficult and Will Remain So; Proceedings of “Structured Programming: The Hard Core of Software Engineering”, a symposium celebrating the 65th birthday of Wladyslaw M Turski, Warsaw 6 April 2003.
- [JacksonM 03b] Michael Jackson; Where, Exactly, Is Software Development? in Bernhard K Aichernig and Tom Maibaum eds, Formal Methods at the Crossroads: from Panacea to Foundational Support; 10th Anniversary Colloquium of UNU/IIST, the International Institute for Software Technology of The United Nations University, Lisbon, March 18-21, 2002; LNCS 2757, Springer-Verlag, 2003.
- [JacksonM 03c] Michael Jackson; Position Paper: A Science of Software Design; position paper for NSF workshop on the Science of Design, Washington DC, November 2-5, 2003.
- [JacksonM 04a] Michael Jackson; Problems, Subproblems and Concerns; Position Paper for the Early Aspects workshop at AOSD 2004.
- [JacksonM 04b] Michael Jackson; Seeing More of the World; IEEE Software Volume 21 Number 6 pages 83-85, November 2004.
- [JacksonM 04c] Michael Jackson; Problem Frames and Software Engineering; Information and Software Technology, special issue on the 1st International Workshop on Advances and Applications of Problem Frames, K Cox, J Hall and L Rapanotti eds, July 2005.

- [JacksonM 05a] Michael Jackson; Problem Structure and Dependable Architecture; in C Gacek, R Lemos and A Romanovsky eds, Architecting Dependable Systems III [Proceedings of IWADS Conference 2004], Springer LNCS 3549, 2005, pages 322-330.
- [JacksonM 05b] Michael Jackson; The Structure of Software Development Thought; in Structure for Dependability: Computer-Based Systems from an Interdisciplinary Perspective, Besnard D, Gacek C and Jones CB eds, Springer, ISBN 1-84628-110-5, 2006.
- [JacksonM 05c] Michael Jackson; The Role of Structure in Dependable Systems: A Software Engineering Perspective; in Structure for Dependability: Computer-Based Systems from an Interdisciplinary Perspective, Besnard D, Gacek C and Jones CB eds, Springer, ISBN 1-84628-110-5, 2006.
- [Jeary 04] Sheridan Jeary and Keith Phalp; On the Applicability of Problem Frames to Web-Based Business Applications; in Proceedings of the 1st International Workshop on Advances and Applications of Problem Frames, 2004.
- [Krogh 98] Birgitte Krogh and Lars Mathiassen; Coordination Problems and Method Features; in Proceedings of the Hawaiian International Conference on Systems Science, 1998.
- [Laney 04] Robin Laney, Leonor Barroca, Michael Jackson and Bashar Nuseibeh; Composing Requirements Using Problem Frames; in Proceedings of the 12th International Conference on Requirements Engineering RE’04, IEEE CS Press, pages 113-122, 2004.
- [Laney 05] Robin Laney, Michael Jackson and Bashar Nuseibeh; Composing Problems: Deriving specifications from inconsistent requirements; submitted to FSE’05.
- [Lanman 02] Jeremy T Lanman; A Software Requirements Engineering Methodology Comparative Analysis: Structured Analysis, Object-Oriented Analysis, and Problem Frames; Department of Computing and Mathematics, Embry-Riddle Aeronautical University, 8 April 2002.
- [Lavazza 04] Luigi Lavazza and Vieri del Bianco; A UML-based Approach for Representing Problem Frames; in Proceedings of the 1st International Workshop on Advances and Applications of Problem Frames, 2004.
- [Li 04] Zhi Li, Jon G Hall and Lucia Rapanotti; Reasoning About Decomposing and Recomposing Problem Frame Developments: A Case Study; in Proceedings of 1st IEEE International Workshop on Applications and Advances of Problem Frames, 2004.

- [Lin 03] L Lin, B A Nuseibeh, D C Ince, MJackson and J D Moffett; Analysing Security Threats and Vulnerabilities Using Abuse Frames; Open University Technical Report No 2003/10.
- [Lin NIJ 04] L. Lin, B. Nuseibeh, D. Ince, and Michael Jackson; Using Abuse Frames to Bound the Scope of Security Problems; in Proceedings of 12th International Requirements Engineering Conference (RE'04), September 2004, Kyoto, Japan.
- [Lin Y 04] Yun Lin; Modelling Abstract Object in Problem Frames; in Proceedings of CSGSC, NTNU, Trondheim, 2004.
- [Nelson 04] Torsten Nelson, Maria Nelson, Paolo Alencar and Don Cowan; Exploring problem-frame concerns using formal analysis; in Proceedings of the 1st International Workshop on Advances and Applications of Problem Frames, 2004.
- [Ourusoff 04] Nicholas Ourosoff; Towards a CASE Tool for Jackson’s JSP, JSD and Problem Frames; in Proceedings of the 1st International Workshop on Advances and Applications of Problem Frames, 2004.
- [Phalp 00] Keith Phalp and Karl Cox; Picking the Right Problem Frame—An Empirical Study, Empirical Software Engineering Journal, volume 5, number 3, November 2000, pages 215-228.
- [Rapanotti 04] Lucia Rapanotti, Jon G. Hall, Michael Jackson and Bashar Nuseibeh; Architecture-driven Problem Decomposition; in Proceedings of the 2004 International Conference on Requirements Engineering RE’04, Kyoto, IEEE CS Press, 2004.
- [Wieringa 04] Roel Wieringa, Jaap Gordijn and Pascal van Eck; Value Framing: A Prelude to Software Problem Framing; in Proceedings of the 1st International Workshop on Advances and Applications of Problem Frames, 2004.
- [YunLin 04] Yun Lin; Applying problem frames to modeling ‘abstraction’ concepts; in Proceedings of the 1st International Workshop on Advances and Applications of Problem Frames, 2004.
- [Zave 93b] Pamela Zave and Michael Jackson; Conjunction as Composition; ACM Transactions on Software Methodology, pages 379-411, October 1993.
- [Zave 97a] Pamela Zave and Michael Jackson; Four Dark Corners of Requirements Engineering; ACM Transactions on Software Methodology Volume 6 Number 1, pages 1-30, July 1997.