Technical Report No: 2003/18

An Example Using Problem Frames: Analysis of a Lighting Control System

Charles B. Haley
Michael A. Jackson
Robin C. Laney
Bashar Nuseibeh

5th December 2003

Department of Computing
Faculty of Mathematics and Computing
The Open University
Walton Hall,
Milton Keynes
MK7 6AA
United Kingdom

http://computing.open.ac.uk

The Open University

An Example Using Problem Frames: Analysis of a Lighting Control System

Charles B. Haley, Michael A. Jackson, Robin C. Laney, Bashar Nuseibeh
Department of Computing
The Open University
Walton Hall, Milton Keynes, MK7 6AA, UK
{C.B.Haley, M.A.Jackson, R.C.Laney, B.A.Nuseibeh} [at] open.ac.uk

***Abstract*—A reasonably complex lighting control system is decomposed using problem frames. The merits of various decompositions are examined. The paper concludes with a discussion of unresolved problem concerns exposed by the decomposition.**

***Index Terms*—Problem Frames, decomposition, requirements, specifications.**

I. Introduction

Problem frames [4] are used to decompose larger problems into a set of smaller ones. The process continues until the smaller problems fit into an understood problem category, or problem frame. The resulting individual frames are analyzed, and then recomposed into a solution for the original problem.

The purpose of this paper is to explore how a reasonably complex problem might be decomposed, then to look at some of the questions the decomposition leaves us with[1]. The paper describes what might best be considered a learning exercise, drawing on previous work [3, 4] as well as [2] and [1]. The chosen problem was a lighting control system. The scenario is designed to explore certain difficulties, such as security and concurrency.[2] It is not intended to present a 'real-life' analysis of a control system.

This paper is structured as follows. Section 2 presents the original requirements statement for a lighting control system as received from the customer. Section 3 is an exercise in clarification, exploring using separation of the requirements into *optative* and *indicative* statements. Section 4 presents the resulting context diagram. Section 5 explores decomposing the problem. Section 6 examines some questions that recomposition must answer, section 7 looks at specific concerns raised by the decomposition, and section 8 concludes.

II. The Problem Statement

The lighting control system to be built must conform to the following general requirements, provided by the firm constructing the building.

The architect wishes to have a lighting control system for a building. From the user's perspective, the system consists of switches and lighting units (lights) associated with a room. When a switch is actuated, the associated light or lights in the room must be turned on or off.

The architect requires the use of up/down momentary contact switches. A momentary contact switch must cause its lighting units to be in the state indicated by the switch's motion: up turns the lights on if they are not already on and down turns the lights off if they are not already off.

The system must include a master control panel that indicates the state of the lighting units in each room. If the lights are on in a room, the indicator on the panel shows green. If the lights are off, the indicator does not glow. The state of the lights in any room can be changed using the panel. Only certain people are allowed to use the control panel; they must identify themselves using a proximity badge (see below) that confirms their identity.

Lighting units contain a unique identifier. The system must keep track of where each lighting unit is (which room it is associated with) and how long any given lighting unit has been illuminated.

Certain lights are in secure rooms and are to be actuated only by people with an appropriate level of authorization. Users carry an identity card (a proximity badge) that is read by a proximity reader either embedded in or installed next to a switch. Lack of a card means the person has the lowest level of authorization possible. The level of security necessary for a room is established using the master control panel. The system must record who operated the lights in a secured room. A person who lacks authorization may not change the state of the lights.

The owner of the building requires the system to be able to trace all light on or light off actions, printing the trace in real time on a printer in the control room. If this printer is not working correctly, an alarm of some kind must be given.

The system must monitor the lighting units. If a lighting unit is not in the correct state (e.g. off when it should be on, or not responding at all), the system must try to correct it. If the correction fails, the system must indicate this fact by changing the indicator on the master control panel of the room containing the failing lighting unit to show red and logging the printer discussed above. The detection of a lighting unit not associated with any switch is to be logged on the printer and indicated by illuminating a red indicator on the master control panel reserved for this eventuality.

[1] The work described in this paper was done during December 2002 – March 2003

[2] Although developed independently, the scenario resembles one found in [6]. The major differences are multiple control interfaces, incorporation of security requirements, and dynamic definition of 'rooms' for control purposes.

III. Clarifying The Requirements

Our first problem is to clarify the requirements. As part of this experiment, we choose to proceed by separating the requirements into indicative statements (statements about how the world is) and optative statements (statements about how we want the system to behave). The process will naturally provoke a certain number of questions, this format is chosen to present the information. Some of these questions and their respective answers are given below.

A. Indicative Statements

The indicative statements are (some of are implied):

- Lighting units can be turned on and off.
  - *Can one read the state of a lighting unit? Yes.*
  - *What is their reset state? Off.*
  - *How does one send a command to a unit? The units are on a network. One writes a command to the network. If a lighting unit recognizes its ID, then it takes appropriate action and responds with its current status.*
  - *Is there a "Get Status" command? Yes.*
  - *Are the lighting units robust? Yes. Telling a lighting unit to turn on while on and vice versa has no effect other than to produce a status packet.*
- Lighting units contain a unique identifier.
  - *What is its format? It is a 32-bit number.*
  - *How does one get the identifier of an unknown unit? A lighting unit that is under power will attempt to register with a "name service". One can find the list of unused lighting units by comparing the list of registered units with the list of known units.*
- The system contains proximity readers (one that can read badges in its vicinity)
  - *How are the badge readers connected? Like the lighting units, the readers are on a network.*
  - *How does a reader report a badge? By sending a report when a badge enters or leaves its range.*
  - *What information does a reader provide? It provides a unique ID for the reader (a 32-bit number), and the number of the badge that has entered or left its sensing area.*
  - *What is a badge number? A badge number is a 12-digit decimal number. Badge numbers are unique across time and space.*
  - *What happens if two people are standing near a reader? Both badges will be detected and reported.*
- There is a master control panel. It contains a badge proximity reader.
  - *Is the master control panel something purchased, or is it to be designed as part of the system? It is purchased.*
  - *How is the panel communicated with? Like the lighting units and the proximity readers, the master panel is connected to a network.*
  - *How does it work? It contains rows of indicators with associated buttons. The indicators can glow green, red, or both (resulting in yellow). The buttons are momentary contact pushbuttons. The indicators are addressed by their [row, column] position. The buttons send the events down(r,c) and up(r,c). Button events can be interleaved (down(1,1), down(1,2), up(1.1), up(1,2)). The panel is robust, in that no series of commands will cause it to enter an unknown state.*
  - *What about its initial state? All indicators are off and all buttons are up. The panel has a special button (-1,-1) that is pushed when the panel is turned on (given power). The down/up events for this button will never be interleaved with any other event.*
- The system uses up/down momentary contact switches.
  - *What do the switches switch? Nothing. They send signals to some listening computer.*
  - *How to they identify themselves? The have a unique ID, which is a 32-bit number.*

B. Optative Statements

The optative statements are:

- Lighting units are associated with rooms.
  - *What is a room? An arbitrary collection of lighting units. In other words, a 'logical room'.*
  - *Can a lighting unit be associated with more than one logical room? No.*
  - *Can 'physical rooms' contain more than one "logical room"? Yes.*
  - *Can 'logical rooms' span 'physical rooms'? Yes. There is no relationship between physical rooms and logical rooms. One can impose limits on the span of a logical room (e.g. across multiple buildings).*
- Switches are associated with rooms.
  - *Can a switch be associated with more than one logical room? Yes. A switch can be something like a 'master switch' that controls lights in several rooms.*
- When a switch is lifted, the controlled lighting units are turned on. When a switch is lowered, the controlled lighting units are turned off.
  - *What is to happen when the lighting units are already in the correct state? Nothing. This is a non-event.*
- The master control panel must indicate the state of lighting units in each room.
  - *How is the room indicated on the panel? The indicator and button at R,C are associated with a room. The association is arbitrary, and is set up by hand. The associations are written on the panel using special labels.*
- If the lights are on in a room, the [associated] indicator on the panel shows green. If the lights are off, the indicator does not glow. […] The system must indicate [a failed lighting unit] by changing the indicator on the master control panel of the room containing the unit to show red.
  - *What is to display if the lights are to be on but one has failed? Both red and green, which results in yellow.*
  - *What is to display if the lights are to be off, but one has failed? Red.*
  - *What is a failure? A lighting unit reporting a state inconsistent with its last command, or failing to*

report at all.

- The state of the lighting units in any room can be changed using the [master control] panel.
  - *How? Pushing the button associated with a room causes the lights in the room to invert their state.*
- The amount of time any given lighting unit has been illuminated must be kept.
  - *Does the lighting unit keep track of the information, or must the system act as a proxy? The lighting unit does not keep track of the amount of time it has been illuminated.*
- A proximity reader can be either embedded in or installed next to a switch.
  - *Is the reader actually in the switch? No, it is merely co-located.*
  - *Is the difference between "in" and "next to" significant? No.*
  - *Do all switches have readers? No. Absence of a reader means that a badge is not necessary.*
  - *How does one know which reader(s) are near which switches? This must be determined manually.*
  - *If a reader is near several switches, does it control all of them? It should be able to, if appropriately parameterized.*
- Users might carry an identity card readable by the badge proximity readers. Lack of a card means the person has the lowest possible level of security.
  - *How does one associate a person with a badge? There must be a way to connect the badge's unique ID to the person.*
  - *What is to happen when two or more badges are sensed? The badge with the highest privileges behind it is used.*
- In secure rooms, the system must log who actuates the lights.
  - *What is a secure room? It is a room equipped with a badge reader.*
- The level of security necessary for a room is established using the master control panel.
  - *Is this 'must' or 'may'? May. A room is not required to be secure.*
- Only certain people can use the master control panel.
- The system must record who operated the lights in a secured room.
  - *What is to happen when the light is already in the state requested by the switch? The operation is logged.*
- If this printer is not working correctly, an alarm of some kind must be given.
  - *What does "is not working correctly" mean? Any alarm the printer gives back, such as out of ink or out of paper, or failure to respond to a status enquiry.*
  - *What is the alarm? An audible alarm is to sound outside the control room and anywhere else that the architect specifies.*

IV. The Context Diagram

The above problem statement and discussion lead us to propose Figure 1 as the context diagram for the system.

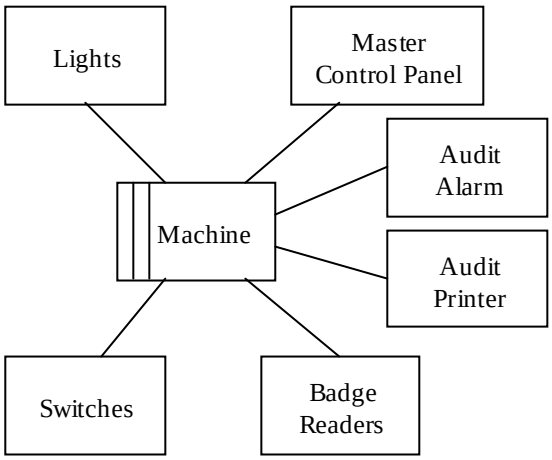

Figure 1 – An Incomplete Context Diagram

This diagram mentions all the components of the system listed in the problem statement and relates them to the machine. Unfortunately, the diagram leaves out several important parts of the problem. In particular, when a person actuates a switch, it is the person who may be carrying a badge. The badge identifies the person to the system, and establishes the person's privileges. The privileges determine whether the switch actuation is to be honored. Therefore, the person, the badge, and the privileges are important parts of the problem and should be included in the context diagram. After doing so, we have the following diagram:

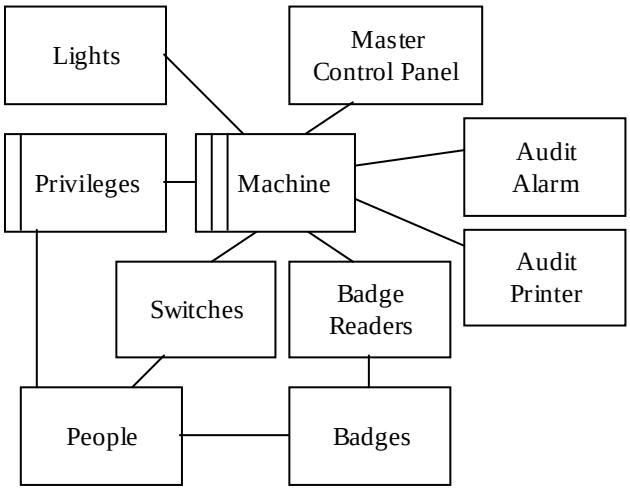

Figure 2 – The Completed Context Diagram

V. Problem Frame Diagrams

A. Initial Thoughts

There is nothing physical that relates a switch to the lights it controls or to the logical room that contains the lights. Equally, there is nothing physical that relates a badge reader to a switch or to a room, or relates a badge to a person. It seems that the notion of *room* is a unifying concept fundamental to the problem, and perhaps the problem could

be decomposed along that dimension.[3]

Actuating a switch is a request that the state of the lights in a room be changed. From the user's point of view (and the switch's as well), the lights in a room are treated as a unit. It makes sense, therefore, to incorporate the notion of *room* into the switch phenomena along with the *up* and *down* phenomena. A method to map switches and lights to rooms is required. Following this line of reasoning further, it becomes clear that the badge and privilege determination are separate from the switch actuation. A badge is associated with a person and privilege is associated with a person/room pair, meaning we need another map. We thus end up with the lexical domains *People → Privs*, *Switches → Rooms*, and *Rooms → Lights*.

One of the fundamental problems, controlling the lights, seems to be a *commanded behavior* problem. The people are commanding the lights using the switches and the master panel. However, it would seem that the master panel presents enough differences from use of the 'normal' switches to justify separating the two into distinct subproblems, *Switches & Lights* and *Master Control Panel*. The next problem considered is the *Audit* problem, which responds to the parts of the problem statement requiring verification that the lights are in the state that they should be. The last problem is the maintenance of the lexical domains.

Please note: the phenomena at the boundaries of the domains are not given in the following diagrams. As will be noted later, they should be.

B. The Switches & Lights Problem

Accepting this first analysis, the problem frame diagram for the switches problem could be:

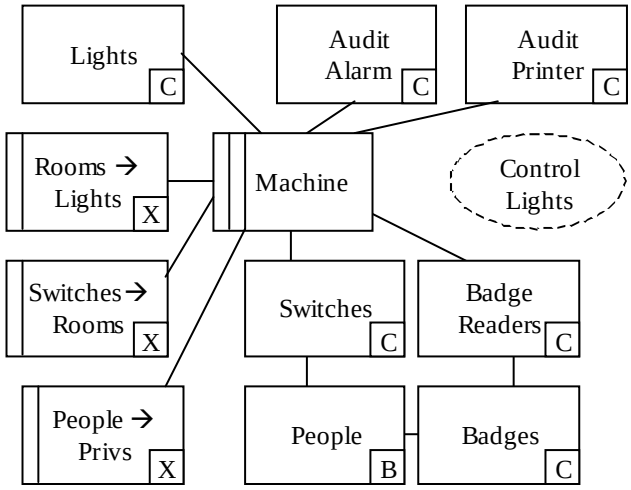

Figure 3 – An Overly Complicated Solution

Unfortunately, this problem frame diagram is far too complex to be of use. It hides multiple requirements under the name "Control Lights". It doesn't fit any basic problem frame. For these reasons and others, it is not worth trying to complete the requirements arrows or frame concern. We need to subdivide the problem further.

We start by connecting the switches to the lamps in the rooms that they control. This is a commanded behavior problem. The requirement, roughly stated, is *If the user actuates a switch, then the lights in the room(s) associated with the toggle should turn on, turn off, or remain as they are, depending on the actuation.* The problem diagram would look something like:

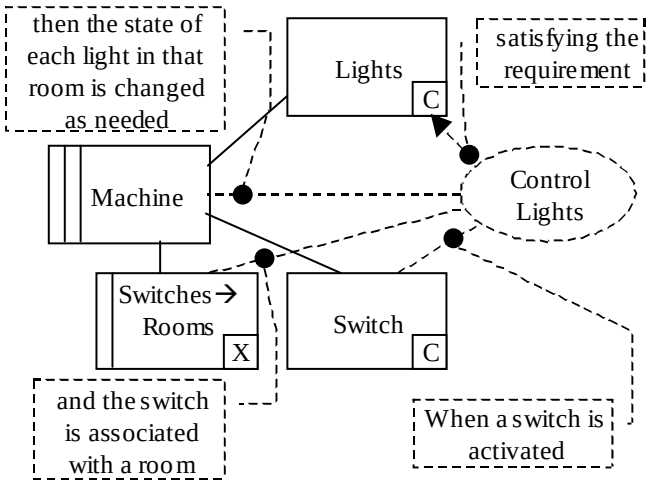

Figure 4 - Basic Lights Control

We now turn our attention to the security aspects of the problem. The requirement is, again roughly stated, *If a room is secured, then only people with the appropriate permission can cause a state change in the lights. People are identified by badges.* This is a required behavior problem, with a diagram that looks something like (almost all detail has been intentionally omitted):

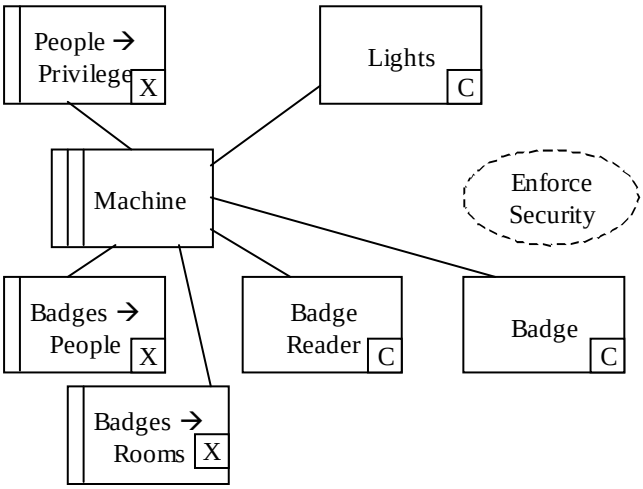

Figure 5 – A First Attempt at Security

This problem diagram suffers from the same fault as Figure 3; it is too complex to reason about. For example, according to the information supplied, badge readers note when a person enters and exits its detection area. This behavior is not made explicit in the diagram, and it is difficult to do so without adding secondary requirements.

We can reduce the complexity of the diagram by introducing a model for *Person in Room*. The enter and exit events generated by the badge reader give us the information we need to build the model. A person is considered in a room between the enter and exit events. The

[3] We will determine later that this assumption is incorrect.

model is used by a second subproblem that verifies permissions and enforces security. Following this route, we find we have two problems, one to build the Person/Room model and one to use it.

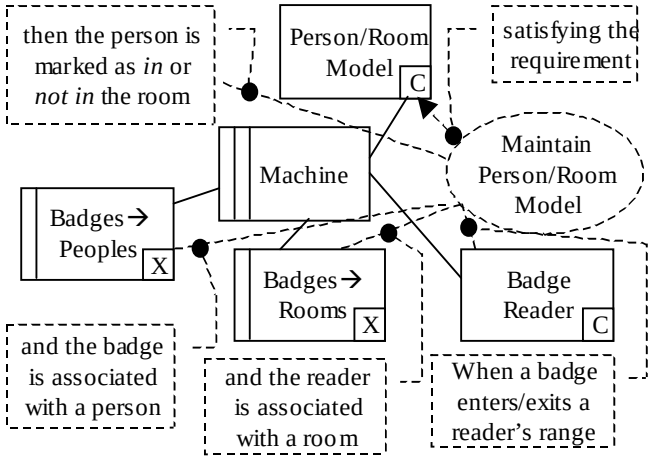

Figure 6 – Security: Building the Person/Room Model

Using the model would seem to be straightforward. The required behavior problem would be similar to the switches commanded behavior problem illustrated in Figure 4. However, the resulting diagram would yet again suffer from being overly complex and needs to be broken into two subproblems. To save space, only the resulting subproblems are presented here.

The first subproblem diagram, named *Control Lights* is the same as Figure 4 except that the controlled domain has been changed to a different subproblem named *Lights in Room*. The only difference between the two is the phenomena: the *Lights in Room* subproblem expects events of the form **up(Room)** and **down(room)** instead of **on(Light)** etc.

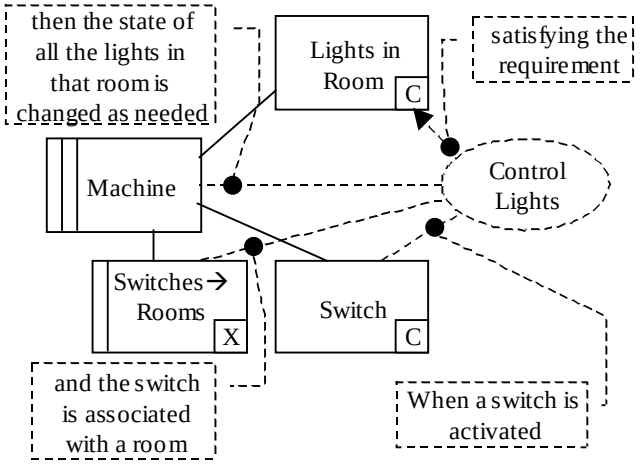

Figure 7 – Control Lights

The next diagram shows the *Enforce Security* required behavior problem.

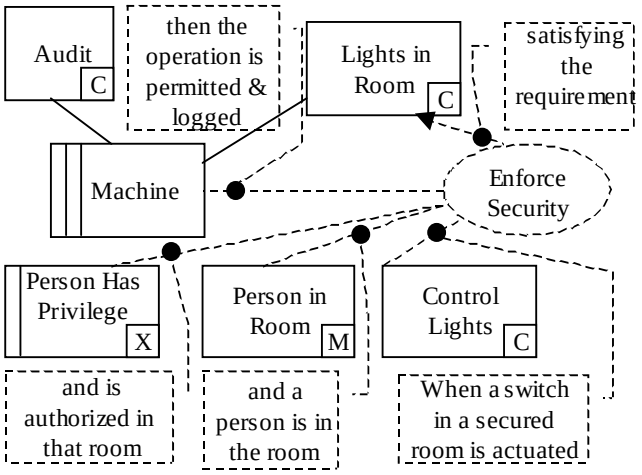

Figure 8 – Enforce Security

Another required behavior problem diagram is needed to complete the security solution. The initial problem statement specifies that all attempts to turn lights on or off that are rejected for security reasons must be logged. The subproblem diagram is very similar to *Enforce Security* and will not be included here. It is a separate subproblem to avoid having "or" conditions in the *Enforce Security* requirement.

We complete the picture with the diagram of the *Lights In Room* commanded behavior problem. Its requirement is approximately *when a switch in a room is raised, then the lights in that room are to be turned on. If a switch in a room is lowered, then the lights in that room are to be turned off.* The diagram is:

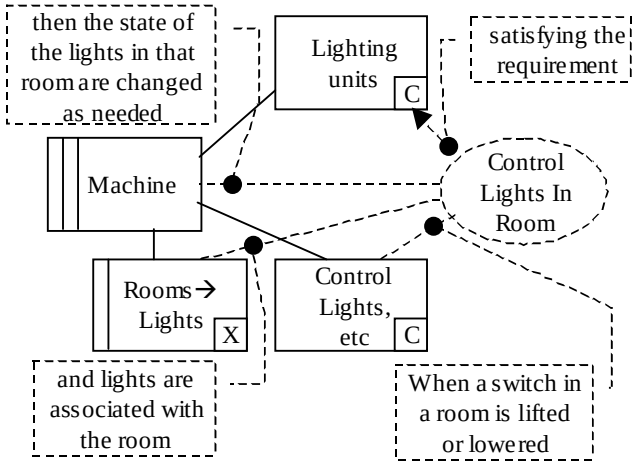

Figure 9 – Lights in Room

C. The Master Control Panel

The Master Control Panel subproblem is decomposed into four subproblems. The first is an information display problem in which the indicators are set appropriately. The second is a commanded behavior problem, where pushing a button associated with a room inverts the state of the lights in that room. The third is a commanded behavior problem that actually does the inversion of the state. The fourth is the security problem, which is a required behavior problem.

The diagram of the information display subproblem is:

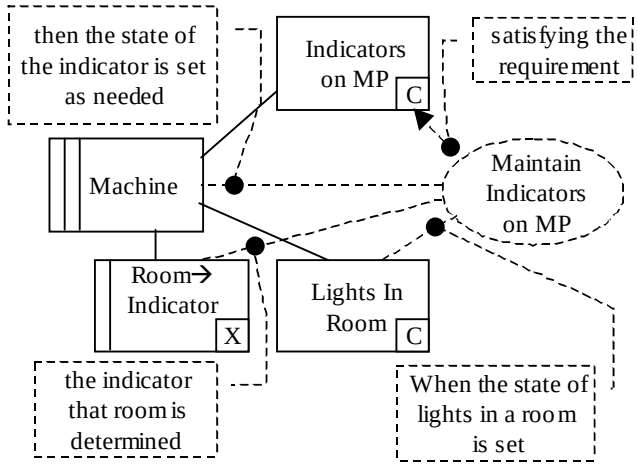

Figure 10 – Master Panel Indicators

The diagram of the second subproblem, controlling the lights from the master panel, is:

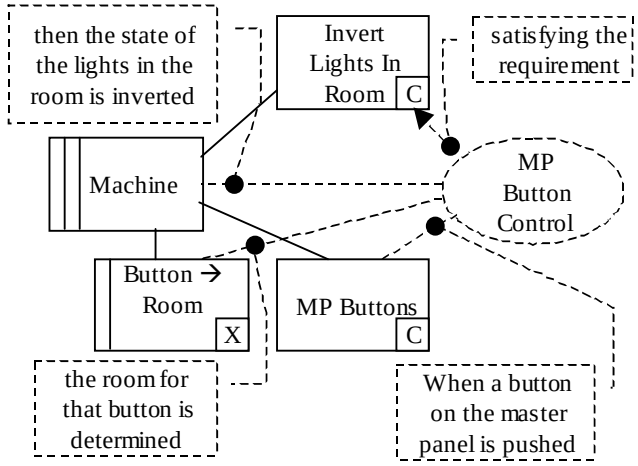

Figure 11 – Master Panel Buttons

The third problem, *Invert Lights in Room,* is identical to *Lights in Room* except that a lexical domain is used to keep track of the last commanded state of the lights in the room so that it can be inverted. The diagram is:

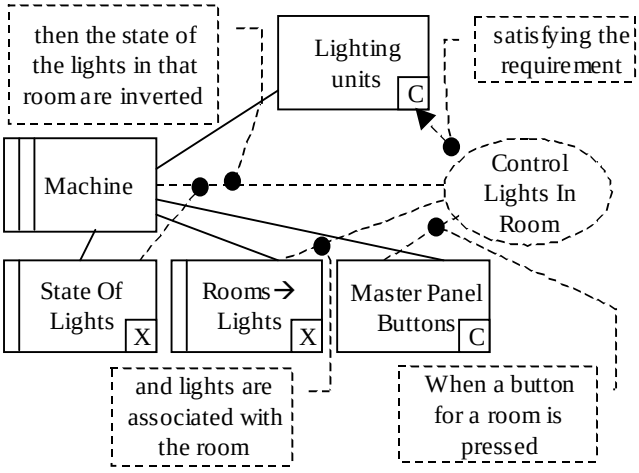

Figure 12 – Invert Lights in Room

The fourth problem is very similar to the switches security problem, and is not shown here.

D. The Audit Subproblems

The Audit problem is decomposed into two required behavior subproblems and one information display subproblem. The first scans the lights in each room to determine if they are in the proper state. The second lights the fault indicator on the MP if the first finds a fault. The information display subproblem controls the audit printer and the audit alarm(s).

The diagram for the first problem is:

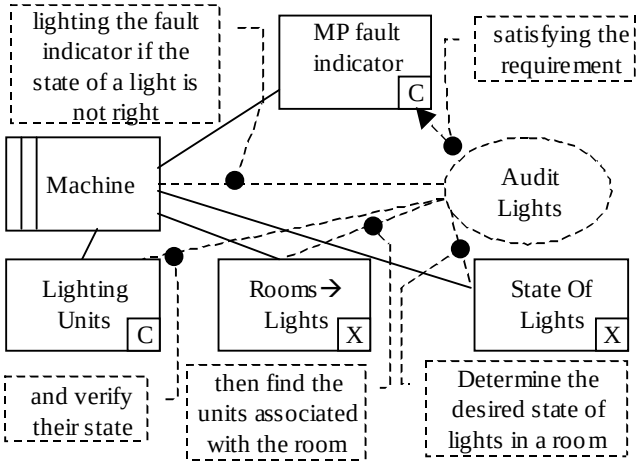

Figure 13 – Audit Lights

The second subproblem, *MP Fault Indicator*, and the third subproblem, *Display Audit Trail*, are very similar to subproblems already presented, and are not shown here.

E. The Lexical Domains

Several lexical domains have been used in the above diagrams. The creation and maintenance of each of these is described by a simple workpieces frame. The subproblems are very similar, so only one example will be provided here.

The *Rooms → Lights* simple workpieces subproblem diagram is:

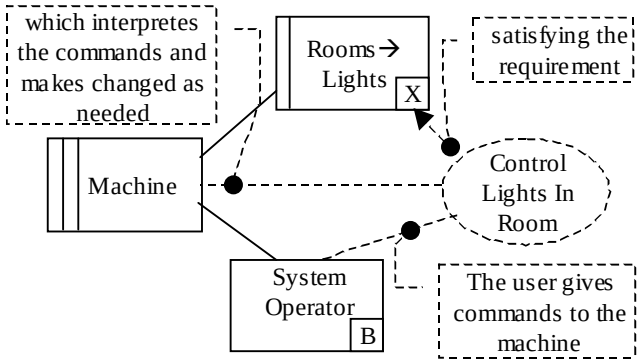

Figure 14 – Maintain Room → Lights Lexical Domain

VI. RECOMPOSITION

One must now recompose the subproblems into a solution to the original problem. Doing so raises the following concerns.

A. Verifying the projections

Each subproblem is a (probably incomplete) projection of the context. It is interesting to note that the context diagram itself appears to be a projection of something left unsaid, as many designed domains in the problem diagrams do not appear in the context diagram.

One could argue that any designed domain that appears in more than one problem diagram should also appear in the drawn context diagram. The rationale is that any domain that appears in only one problem diagram resolves some internal concern. One is tempted to assert that *designed domains that resolve internal concerns and appear in two or more projections should be included in the context diagram*.

B. Verifying the phenomena

Phenomena of one projection that are referred to by another projection (another subproblem) must be sourced or sunk by a domain in the first projection. This assertion raises several naming problems; as the name of the phenomenon is the connection, the names must be unique.

This verification process must extend throughout the virtual context, ensuring that all 'sunk' phenomena have a source and that the parameters of the phenomena at their sources and sinks are consistent.

C. Verifying the connections

The set of subproblems *Audit Lights*, *Invert Lights in Room* and *Lights in Room* illustrate a problem that seems hard to detect automatically. *Audit Lights* depends on the existence of the lexical domain State of Lights, which is maintained by *Invert Lights in Room*. However, State of Lights is not maintained by *Lights in Room*, which means that *Audit Lights* will not function properly. It would be nice to have a way to automatically detect this kind of anomaly. In any event, maintenance of the State of Lights domain must be added to *Lights in Room*.

D. Composition-created domains

There are cases where the act of composition needs to create domains. As Michael Jackson said so succinctly to Charles Haley [5] in response to questions during the exercise, *whenever there is a requirements conflict of [a] "do it / don't do it" kind and the "don't do it" requirement takes precedence, it is necessary to find a way of breaking the "do it" causal chain. One standard way is to split apart what would otherwise be regarded as a shared phenomenon, interposing a machine that can either complete or break the causal chain.* Interposing a new domain could create new phenomena, could mandate changing the names of the phenomena so that verification can take place, and certainly changes the source & sink domains of particular phenomena. Should these new domains be included in the projections (the subproblems) that caused their creation? Should a separate projection be made that reflects the existence of these created domains? These questions are not yet answered.

E. Distribution

In theory, the machine in each subproblem could be a separate computer. In practice, this will not happen. It would be very nice to codify the cases that force merging of the machines. For example, the existence of shared state could force merging. Does this mean that if two machines source the same symbolic phenomenon, they must be combined?

As phenomena are in the end 'shared', one could argue that distribution is *never* allowed because it breaks the simultaneity assumptions of problem frames analysis. Ignored connection domains create similar difficulties. In the same comments mentioned above, Jackson [5] says that *guards to be evaluated in one subproblem could be added to events in another subproblem. This solution method is particular to one kind of composition and to a special (undistributed) kind of problem environment*. This comment clearly states that certain compositions force a non-distributed implementation.

If the simultaneity properties of an interface were made explicit, then many of these questions would disappear. In addition, a more rigorous verification might be possible.

F. Concurrency

The notion of trying to detect potential concurrency problems during composition is intriguing.

Concurrency problems exist on at least two levels. The first is rather large, exemplified by lexical domains and models. There is an inherent concurrency problem between a machine that maintains the domain and a machine that uses it. The problem manifests itself as inconsistent or partial state. It would seem that this sort of problem is amenable to analysis, at least at the phenomena level, by applying transaction semantics to the phenomena.

The second level can be illustrated by looking at the example presented in this paper. It is perfectly permissible to have multiple switches for the same room. The switches and lights in a room may not be controlled by the same computer. As such, the nature of the concurrency problem depends on how the system is distributed.

VII. Specific Concerns

Many concerns arise because of problem recomposition or conditions outside the analysis. These are called *specific concerns* in [4].

A. Initialization

Some of the initialization concerns might be:

1) What happens after a power fail?

What is the system supposed to do when power is applied, either for the first time or after a power fail? Is the building to remain dark, or are the lights restored to their previous state? As an example of what might come out of a discussion of this form, we might discover that lighting units have a *safety switch* on them. If the switch is at *safety*, when power is applied to the lighting unit, the light is illuminated. This state is to be maintained until the unit is told otherwise. The existence of this safety switch and what it implies will certainly change several problem diagrams.

2) What about partial power failures, where the controller loses power but the lights don't?

There are several subquestions that might arise while discussing this point. Does a partial power failure trigger a

safety concern? Can power be lost to parts of the control system, and if so what is to occur while power is lost and when power is restored?

3) The audit process cannot run until system is initialized.

This is an example of initialization sequencing. The audit system depends on having the various lexical domains correctly initialized and the lights in a known state. The point after which auditing can start must be determined, and then a required behavior frame added to express the requirement.

4) Lights added to a room are potentially in an incorrect state.

A maintenance engineer may repair or replace a lighting unit while the system is running. Doing so raises concurrency concerns (maintenance of the lexical domains), correctness concerns (the newly installed light is off when it should be on and vice versa), identities concerns (the light may have been moved from another room), etc.

B. Identities

There are many identities concerns. Most of them are recognized by the inclusion of the lexical domains (the → maps). Some, however, cannot be satisfied with the domains. For example, a switch might be added to the system but not associated with any room. A lamp, set to *safety*, might be added to the system but not associated with a room. Badge readers present a similar problem.

Another identities concern, and one that will cause changes to the problem diagrams, comes from the assumption that a) switches are in rooms, b) badge readers are in rooms, and therefore someone in the room is actuating a switch. This assertion is clearly incorrect if there are multiple badge reader & switch pairs associated with a room. We can confuse the identity of the person at the switch with a person in front of another switch for the same room. The solution is to map both badges and switches to a pair *(room, location)* instead of to *room*. The diagram in Figure 6 – Security: Building the Person/Room Model would be changed to build a *Person at Location* model. The diagram in Figure 8 – Enforce Security would be changed to use the *Person at Location* model. Finally, the diagram in Figure 7 – Control Lights would be changed to use a *Switches → Rooms/Location* map.

C. Interference

The decomposition creates several interference or concurrency questions. For example, without care the *Audit* machine can busily undo the *Control Lights* machine's actions. If two switches control the same room and one commands *off* and the other commands *on*, individual lights could be left in conflicting states. Inconsistent states while maintaining the lexical domains is another source of errors.

D. Reliability

The reliability concern touches several of the other concerns. For example, the safety question was discussed above. How the system degrades in the face of power or component failure is another.

VIII. Conclusions

The exercise has shown that problem frames were very useful when reasoning about the structure of the problem. In particular, interactions between requirements (subproblems) were rapidly exposed.

In addition, the exercise brought several issues to the surface:

1. Recomposition of subproblems is non-trivial. Tool support would help by assisting tracking of domains through the various subproblems.
2. One should not take shortcuts with phenomena. One cannot easily reason about the *specific concerns* without the phenomena, which is why most of them are left unresolved. Again, tool support would be very helpful.
3. Some questions about when designed domains should be added to the context. Yet again, tool support would be helpful in ensuring that projections of the context are consistent across all the subproblems.
4. The *specific concerns* can point to changes needed in the subproblems. The analysis is not complete until they are all resolved.

References

[1] Haley, C. B. "Using Problem Frames with Distributed Architectures: A Case for Cardinality on Interfaces" at Second International Software Requirements to Architectures Workshop (STRAW'03), International Conference on Software Engineering (ICSE '03), Portland OR USA, 9 May, 2003.

[2] Hall, J. G., Jackson, M., Laney, R., Nuseibeh, B., & Rapanotti, L. "Relating Software Requirements and Architectures Using Problem Frames," In Proceedings of the IEEE Joint International Requirements Engineering Conference (RE'02). Essen, Germany, 2002.

[3] Jackson, M. *Software Requirements & Specifications*. Addison Wesley, 1995.

[4] Jackson, M. *Problem Frames*. Addison Wesley, 2001.

[5] Jackson, M. Personal communication to Charles Haley: response to questions about problem frames, The Open University, 2003, 19 Jan.

[6] Queins, S., et al. "The Light Control Case Study: Problem Description," *Journal of Universal Computer Science* vol. 6. no. 7, pp. 586-596, Jul 2000.