5

- Other concerns
  - Domain flavours
  - Standard concerns
  - HCI concerns
  - Distribution concerns
- Final exercise

More concerns

- We have analysed, for each problem frame, the typical *correctness* concern, and have provided the **proof structure** for it
- But several other concerns are equally applicable
  - Some of then only apply to certain problem frames, or specific domains, etc.
    - Specific sub-types are called **flavours**
  - We will focus on those that have HCI-related and distribution-related facets

Domain flavours

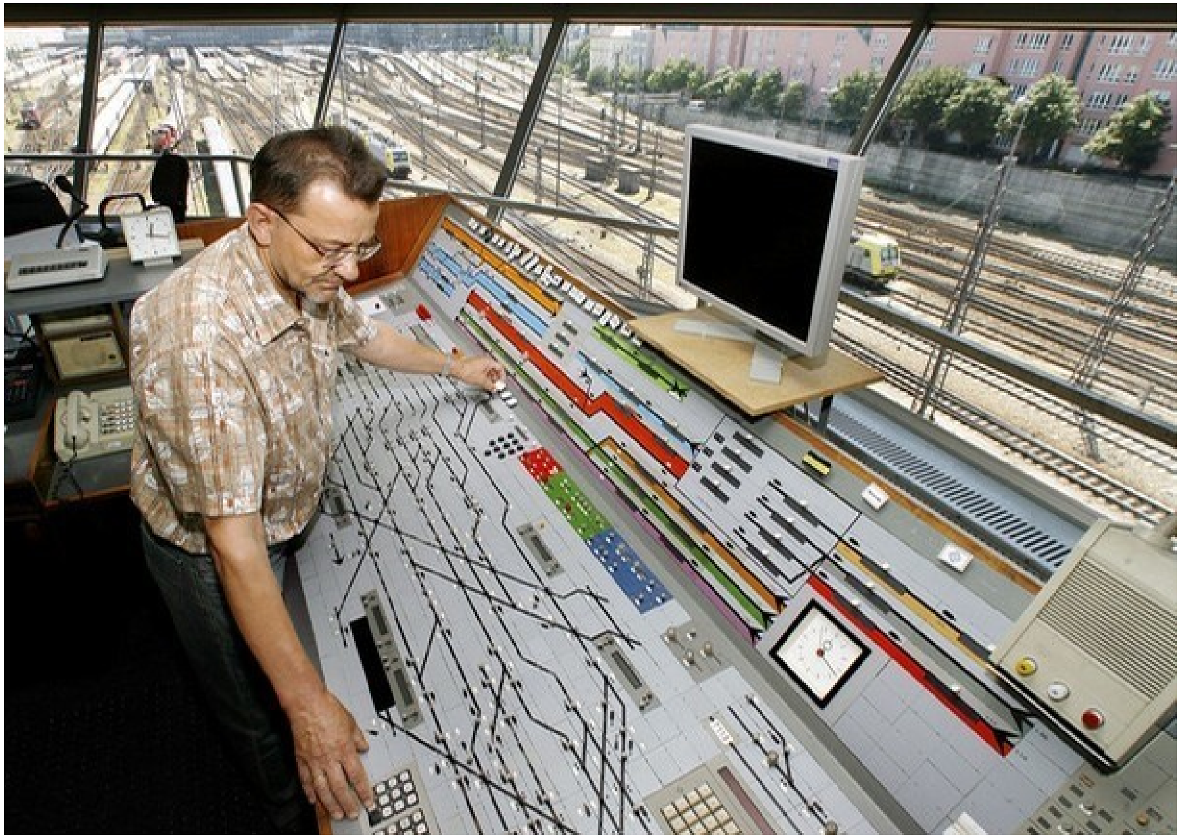
- Static flavours
    - Structurally unchanging (lexical and some causal)
- Dynamic flavours
    - Small-scale behaviour, changes are on a small time scale, have a "resting state"
- Control flavours
    - Large-scale behaviour, changes are semi-permanent
- Informal flavours
    - Cannot perform formal reasoning (causal and biddable)
- Conceptual flavours
    - Missing even a representation, highly subjective

Concerns in static domains

- What is static in a static domain?
  - Values: e.g., an alphabet (static lexical), the road layout of a city (static causal)
  - Structure: e.g., all alphabets are sequences, city layouts are (approximately) graphs
- How static is “static”?
  - Static things may vary, but at a pace lower than that at which things happen in our problem
    - e.g., latin alphabet lost Z and gained G in 3rd century B.C.; then got Y and Z in 1st century B.C.; gained three more letters (*antisigma*, *digamma inversum*, *sonum medium*) in 1st century A.D., and lost them ~60 years later; W came in the middle ages, J and U in the Renaissance

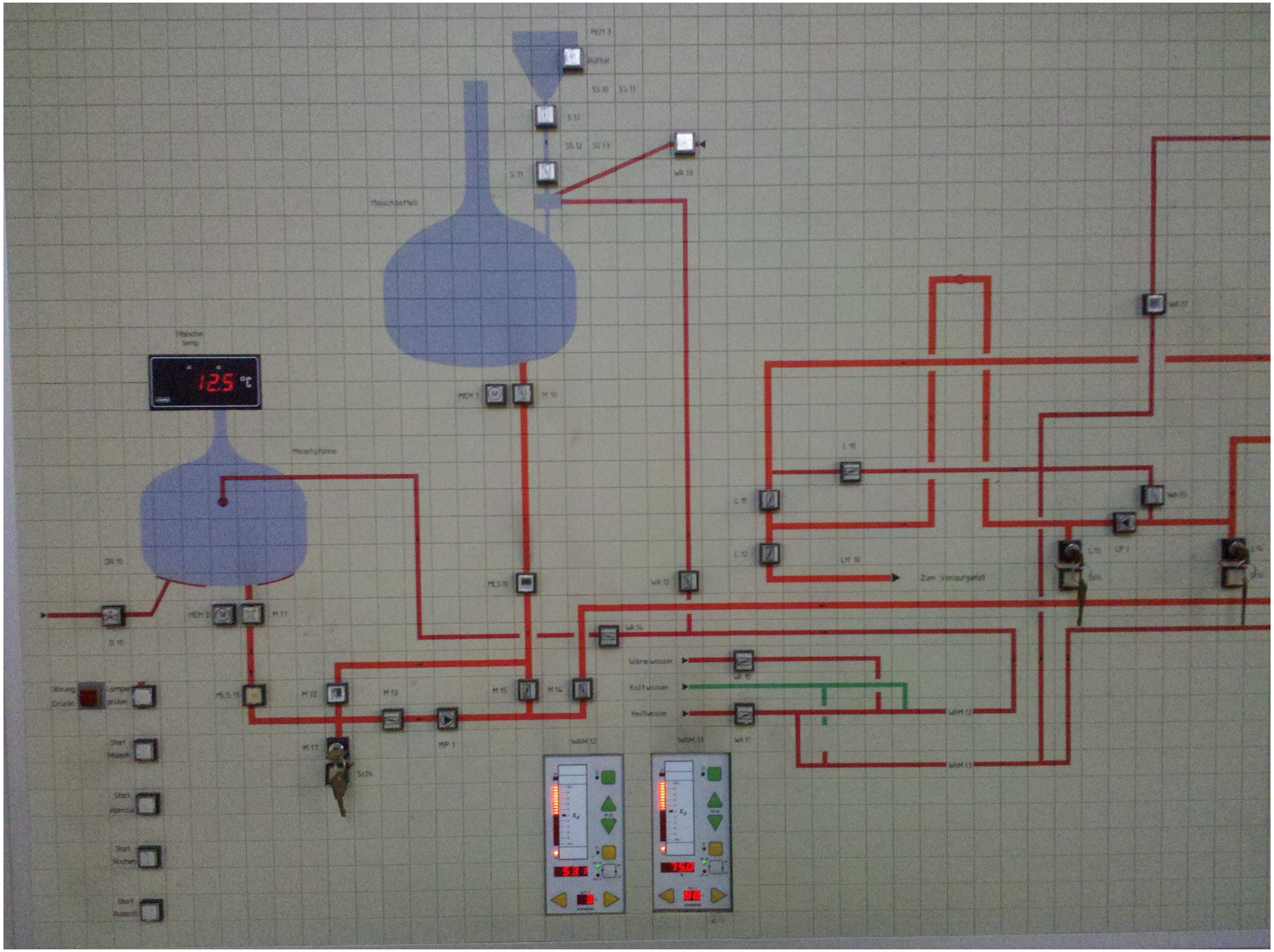
H-concerns in static domains

- It is tempting to present an interface arranged in analogy with the static domain layout
- Just, consider it could change!
- Plus
  - Analogic model
  - Less mapping
- Minus
  - Hard to configure
  - Ergonomics



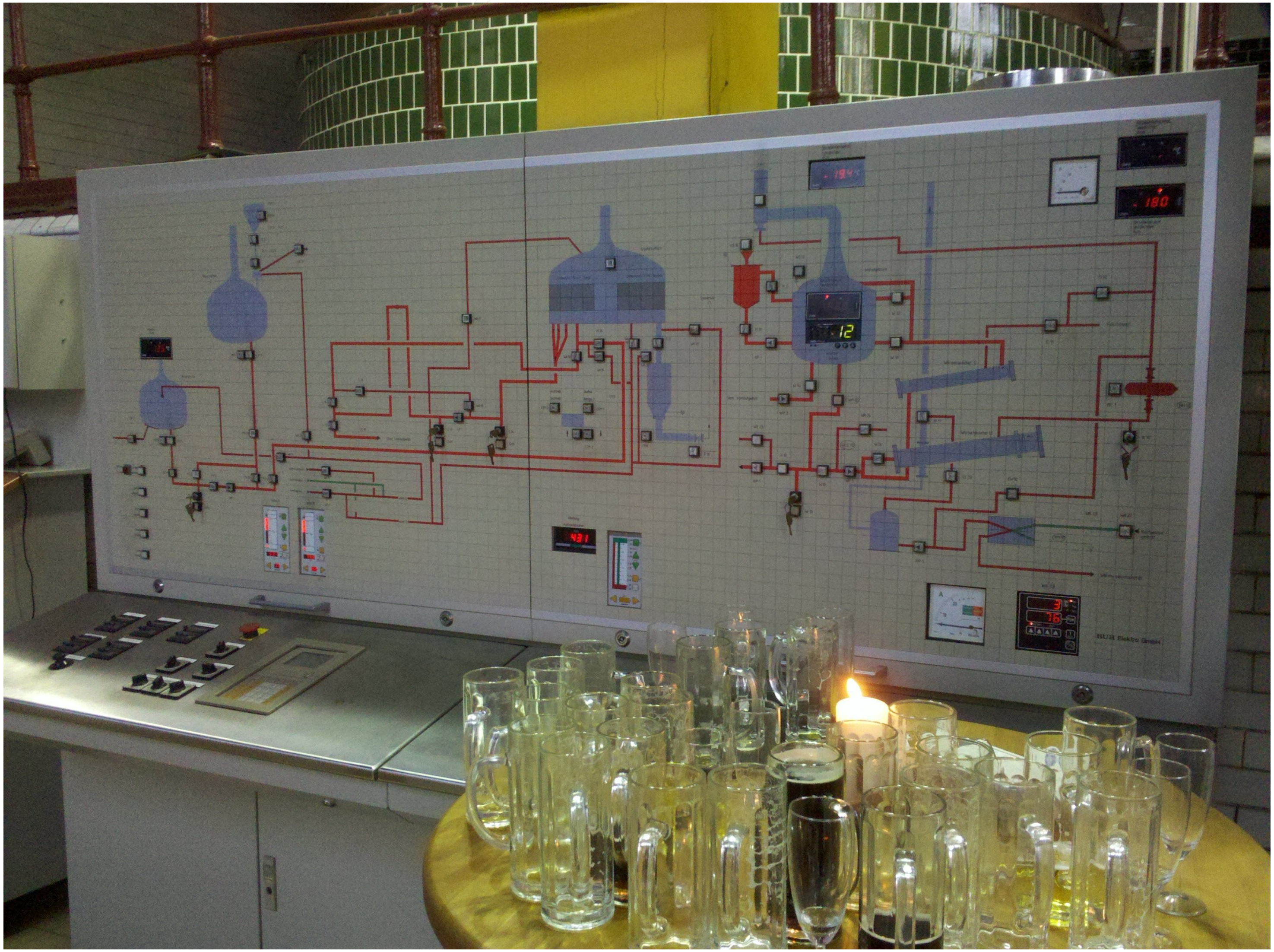
H-concerns in static domains

- Example captured in a recent trip
- Combination of Information Display and Commanded Behaviour
- Guess what this is?



H-concerns in static domains

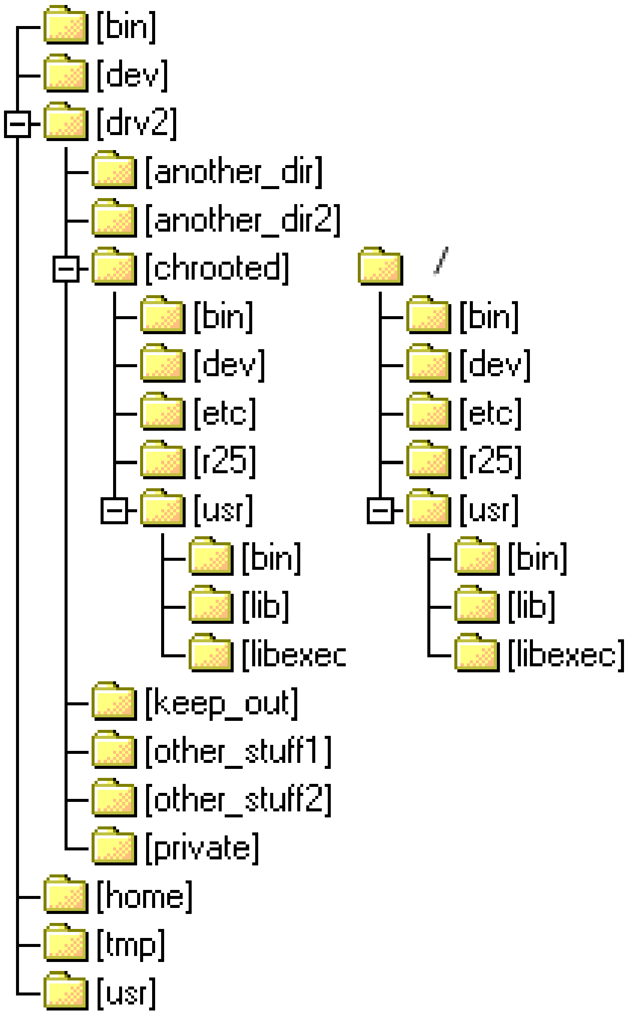
- Example captured in a recent trip
- Combination of Information Display and Commanded Behaviour
- Guess what this is?



Mmmm... Beer!

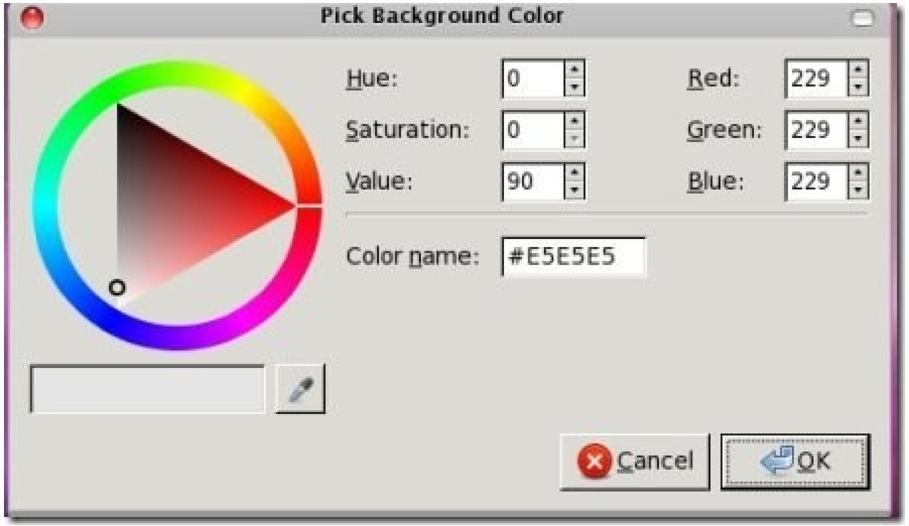
H-concerns in static domains

- Arranging a UI according to the *static structure* is safer (structures rarely change)
- In fact, static structures (but not values) are found in GUI toolkits



```
[bin]
[dev]
[drv2]
    [another_dir]
    [another_dir2]
    [chrooted]
        [bin]
        [dev]
        [etc]
        [r25]
        [usr]
            [bin]
            [lib]
            [libexec
    [keep_out]
    [other_stuff1]
    [other_stuff2]
    [private]
[home]
[tmp]
[usr]
```

```
/
    [bin]
    [dev]
    [etc]
    [r25]
    [usr]
        [bin]
        [lib]
        [libexec]
```



Pick Background Color

Hue: 0
Saturation: 0
Value: 90
Red: 229
Green: 229
Blue: 229
Color name: #E5E5E5

D-concerns in static domains

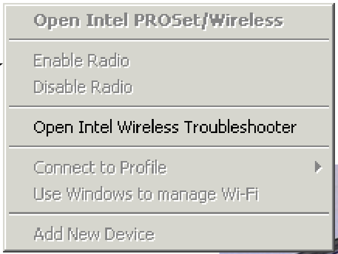
- Static distribution
  - Devices cannot join or leave the distribution infrastructure at will
  - Dangerous assumption when a link could fail and disconnect a device!
- Independent evolution
  - Different parts of a distributed system are often under the control of different entities
  - They can evolve independently
    - an ISP could upgrade bandwidth
    - An end user could install a new browser

Concerns in dynamic flavours

- Tolerance
  - What happens if some external events changes the state of a causal domain?
    - e.g., someone tries to manually force the gates open (or close) in our dam
  - Three possible responses:
    - Robust: events happen, but state will not change
    - Inhibiting: events are prevented from happening
    - Fragile: events happen, state changes to some undetermined one

H-concerns in dynamic flavours

- Tolerance in user interfaces
  - Robust: user can issue inappropriate commands, these are ignored
    - Controlled behaviour frame
  - Inhibiting: user is prevented from issuing inappropriate commands
    - Ghosting out of GUI elements
    - Physical inhibition
  - Fragile: system processes command, goes astray
    - Extremely dangerous!


Open Intel PROSet/Wireless
Enable Radio
Disable Radio
Open Intel Wireless Troubleshooter
Connect to Profile
Use Windows to manage Wi-Fi
Add New Device

PositiveOffset.com

Concerns in dynamic flavours

- Discrete approximation
  - Even when the problem is in the continuos real world, it will end up being treated through discrete approximation by a computer
    - Early or excessive approximation can cause problems
  - Proper way of *studying* the domain might be inherently continuos
    - e.g.: temperature of the water in the dam (hence, volume) over a regular (daily, seasonal, yearly) cycle

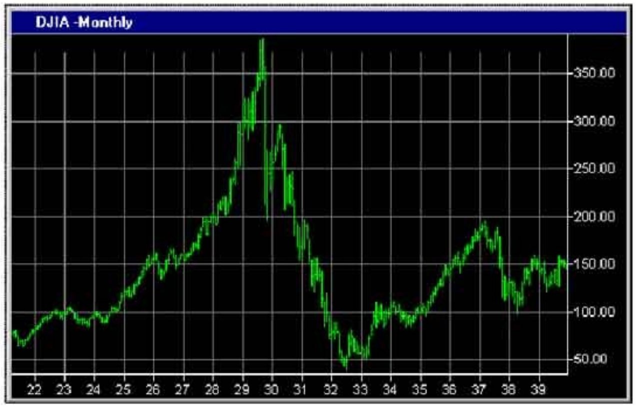
$$\frac{dT}{dt}=k\left(\sin\left(2\pi\omega t\right)-T\right)$$

H-concerns in dynamic flavours

- Discretization in presentation
  - Wrong assessment
  - Confusion
  - Surprise when a value “jumps” to the next discrete step
    - e.g.: value is 1.999 →

Value	1	truncated
Value	2	approximated
Value	2,00	approximated
Value	1,999	real

- Discretization in time
  - Stale data, reading from ages ago and no indication of the fact
  - Insufficient predictability



H-concerns in dynamic flavours

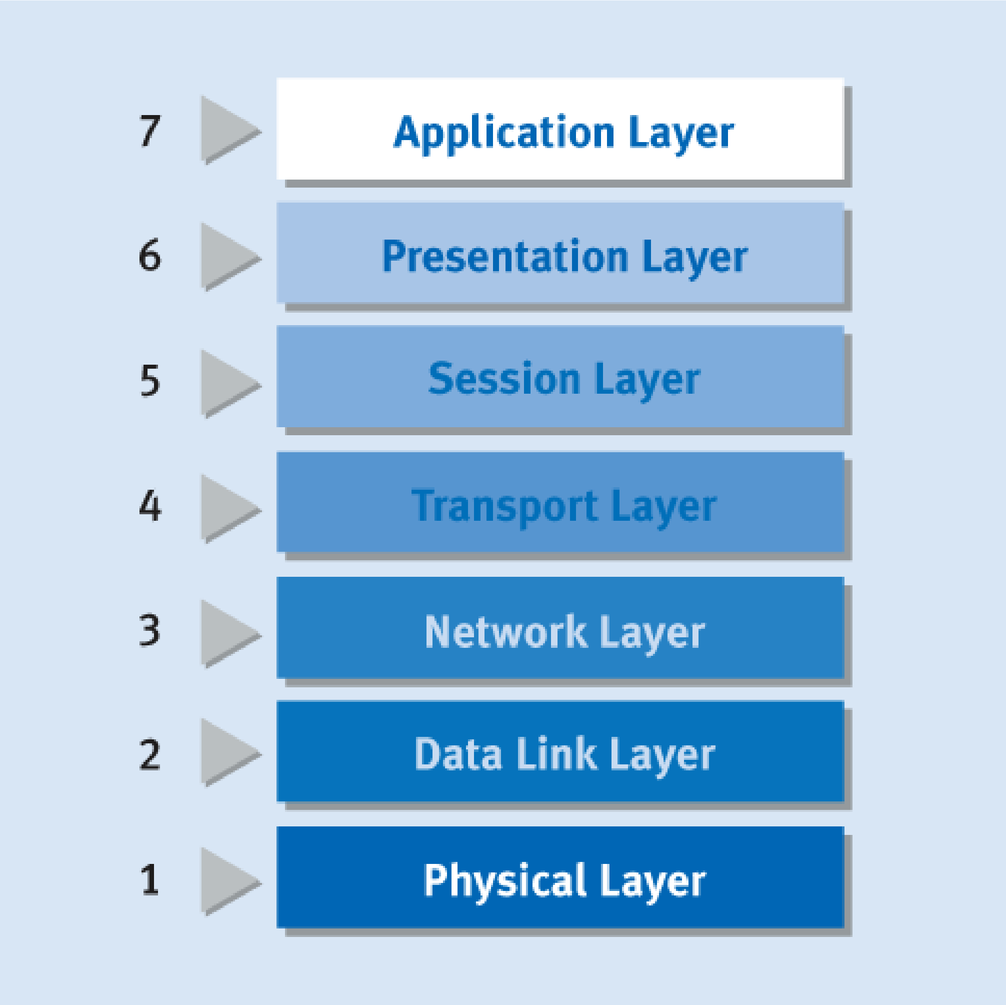
- Discretization in value
  - A continuous input might be forced into a discrete scale
  - Lack of accuracy, frustration
  - Example
    - Slider, 256 positions from “Like” to “Don't like”
    - Radio buttons: “Like” / “Neutral” / “Don't like”

D-concerns in dynamic flavours

- What is the “resting state” of a dynamic flavour domain in a distributed system?
  - Not changing state
    - e.g., don't hear from them / no news is good news
  - Changing state on a regular basis
    - e.g., sending a PING every 100ms
- DSs operate at several simultanous time scales
  - e.g., re-sending a missed packet
  - Hence, multiple levels of dynamicity
    - In a stack model, a layer may appear static on one interface and dynamic on another

D-concerns in dynamic flavours

- Example: ISO-OSI
- The specific job of each layer is to “hide” the complexities of the layer below
- Translate a very dynamic domain into a more quiet one



Concerns in control flavours

- Classification of states
  - Event-active, state-active, pure passive, event-reactive, state-reactive
  - Passive, stoppable-active, unstoppable-active
- In a domain with unstoppable-active state, no phenomenon can interrupt an ongoing transition or processing
  - The machine can find a domain changing with no hope of intervention

H-concerns in control flavours

- In an information display frame on a control-flavour domain with unstoppable-active states, how should the situation be depicted?
  - Modality in UI
  - In MVC model:
    - Controller disabled during unstoppable-active states
    - Model and view updated in “real-time”
  - How to signal transition into and out of unstoppable states?
  - Ask confirmation before causing the domain to enter an unstoppable state?

D-concerns in control flavours

- Unstoppable behaviour a major concern!
  - With **pull** scheme: polling can be suspended
  - With **push** scheme: message might be left waiting
  - With **interrupt** scheme: interrupt must be masked at times (ensure transactions/atomic ops)
    - Plus, very hard to implement properly in general
- Control behaviour of connection domains
  - **Retractability**: can I retract a message that has been sent out, but not executed yet?
  - **Feedback**: will the distribution infrastructure notify the machine of its current state?

Concerns in informal flavours

- Forced formalisation
  - In trying to formalise an informal domain, one could end up in ridicule
  - 'Everybody loves my baby (1), but my baby loves only me (2)'
    - (1) $\forall x$. Loves(x,MyBaby)
    - (2) $\forall y$. Loves(MyBaby,y) $\leftrightarrow$ y = Me
    - (3) from (1), Loves(MyBaby,MyBaby)
    - (4) from (2) and (3), MyBaby = Me
    - All kind of dubious consequences follow...

Concerns in informal flavours

- Wrong formalisation
  - The Three Miles Island case (power plant gone wild):
    - Part of the problem was fitted to an information display frame
    - Requirement: IndicateValveShut $\leftrightarrow$ ValveShut
    - Specification: IndicateValveShut $\leftrightarrow$ SolenoidOff
    - The domain **did not provide** SolenoidOff $\leftrightarrow$ ValveShut
    - But it was formalized (wrongly) as such, so the correctness proof was ok
    - Result...

Concerns in informal flavours

- Wrong formalisation
  - The Three Miles Island
    wild):
    - Part of the problem was
      frame
    - Requirement: IndicateV
    - Specification: IndicateV
    - The domain **did not pro**
    - But it was formalized (w
      correctness proof was c
    - Result...

H-concerns in informal flavour

- Computers cannot handle *informal* input or output
- No hope of interacting on informal phenomena, if not by approximation
  - Is approximate formalization ...
    - Reliable?
    - Satisfactory to the user?
    - Processable?
      - e.g., free form text in a “comments” field

D-concerns in informal flavour

- Our focus is on designing **distributed systems**...
- But we really mean **distributed computer-based systems** with that
  - Hence, we will ignore informal flavours
    - Except for “human” domains
- What cannot be formalized,
  - Cannot be put in a TCP/IP packet
  - Cannot be fed to a CPU
  - Cannot be stored on a data base

Conceptual flavours

- Hard even to consider as physical domains
- Share most of the concerns and h-concerns with the previous ones
- Stay away from conceptual domain if you can!

- We will not discuss them further
  - Epistemology is Monday 14:30-16:00, Wednesday 14:30-16:00, Friday 12:00-13:30
  -

Other common concerns

- **Overrun**
  - Machine too fast or too slow w.r.t. domain
- **Initialization**
  - Establishing the initial state of the domain
- **Reliability**
  - Domain behaves differently from description
- **Identities**
  - Associating related individuals in multiple domains
- **Completeness**
  - What am I missing?

Overrun h-concerns

- Machine too fast for humans
  - Delay cycle
  - Less frequent updates
  - Provide clear feedback
- Machine too slow for humans
  - Prominently display “busy” state
  - Buffer commands / clear buffers (keyboard)
  - Modality in interfaces
  - Inhibit further commands

Initialization h-concerns

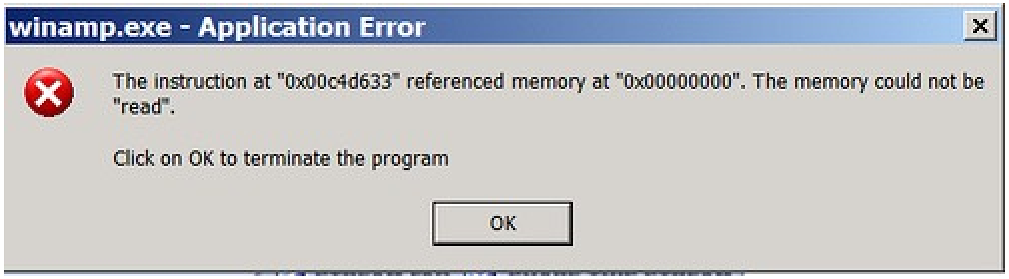
- How to initialize dialogue with a user upon starting up?
  - Let the user knowingly wait
  - Avoid displaying unitialized data
  - Provide visual clue of when data is valid
- How to initialize a controlled domain upon starting up?
  - Ask the user how he/she wants the domain initialized
  - Initialize to a default, safe state (and let the user know)

Initialization h-concerns

- How to handle partial re-initializations?
  - Blackout / poweroff
  - Login, logout
- What if the controlled domain requires user intervention for initialization?
  - User is biddable: instruct on how to initialize the domain
    - e.g., setting up heavy machinery
  - Refuse further interaction until domain initialized properly

Initialization h-concerns

- What if the domain cannot be initialized?
  - e.g., some needed actuator is broken
  - Cannot initialize, cannot proceed: lock-up
    - Enter an explicit “lock-up” state
    - Let the user know what is happening
    - Suggest remedial actions
    - Suggest where to look / whom to call for further help
  - How much detail to provide, which options to give?



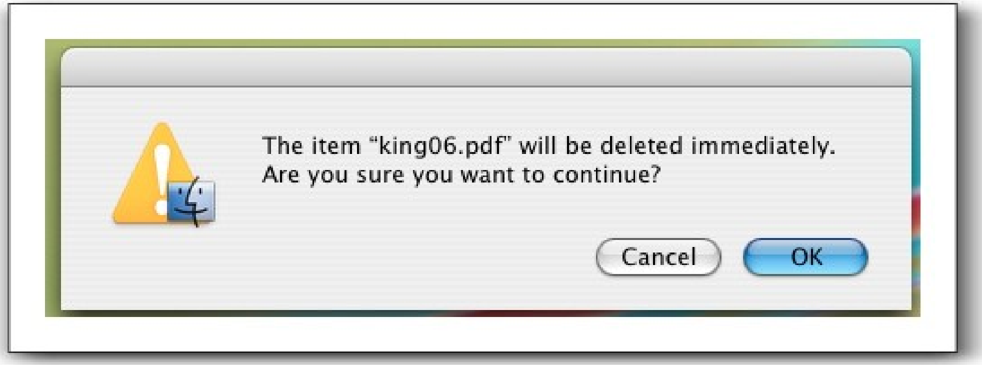
winamp.exe - Application Error

The instruction at "0x00c4d633" referenced memory at "0x00000000". The memory could not be "read".

Click on OK to terminate the program

Reliability h-concerns

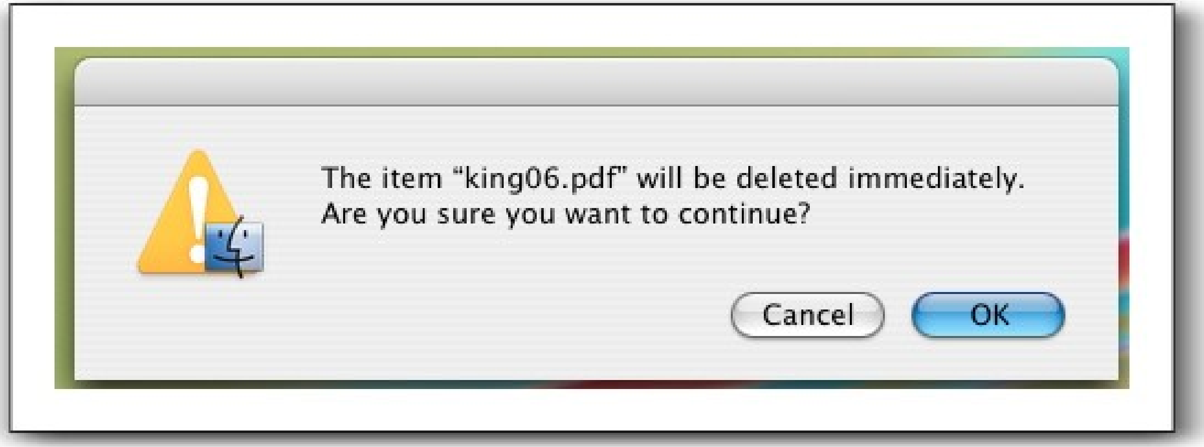
- How to report errors?
  - e.g., syntax errors in lexical domains
- How to diagnose errors in a non-obtrusive way?
  - The user does not want to have his workflow interrupted by “stupid” consistency checks
- Are users “reliable”?
  - Are you sure?
  - Are you sure you are sure?



The item “king06.pdf” will be deleted immediately.
Are you sure you want to continue?

Identities h-concerns

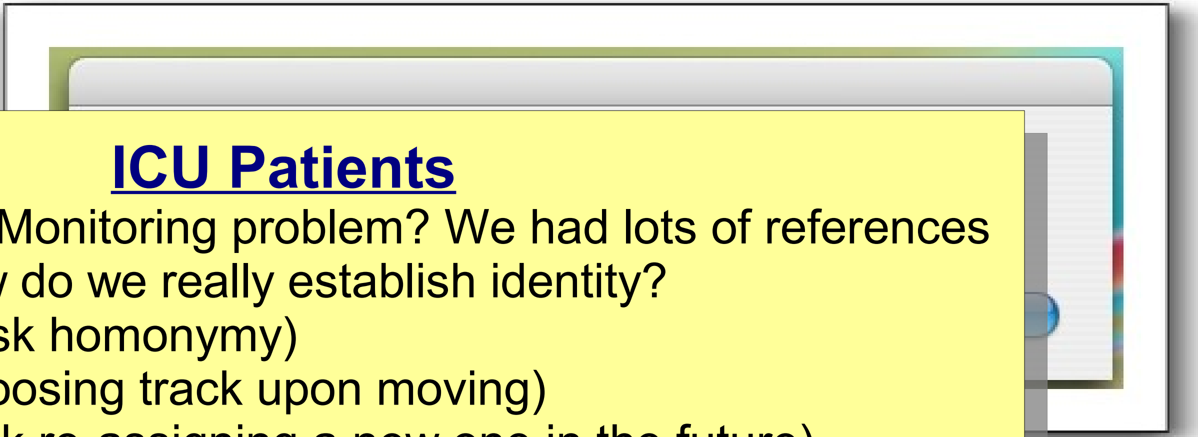
- How to make it clear to the user that different interface phenomena refer to the same individual?
- What if names/labels/IDs are not enough?
  - e.g., files with the same name in different folders



The item “king06.pdf” will be deleted immediately.
Are you sure you want to continue?

Identities h-concerns

- How to make it clear to the user that different interface phenomena refer to the same individual?
- What if names/labels/IDs are not enough?
  - e.g., files with the same name in different folders



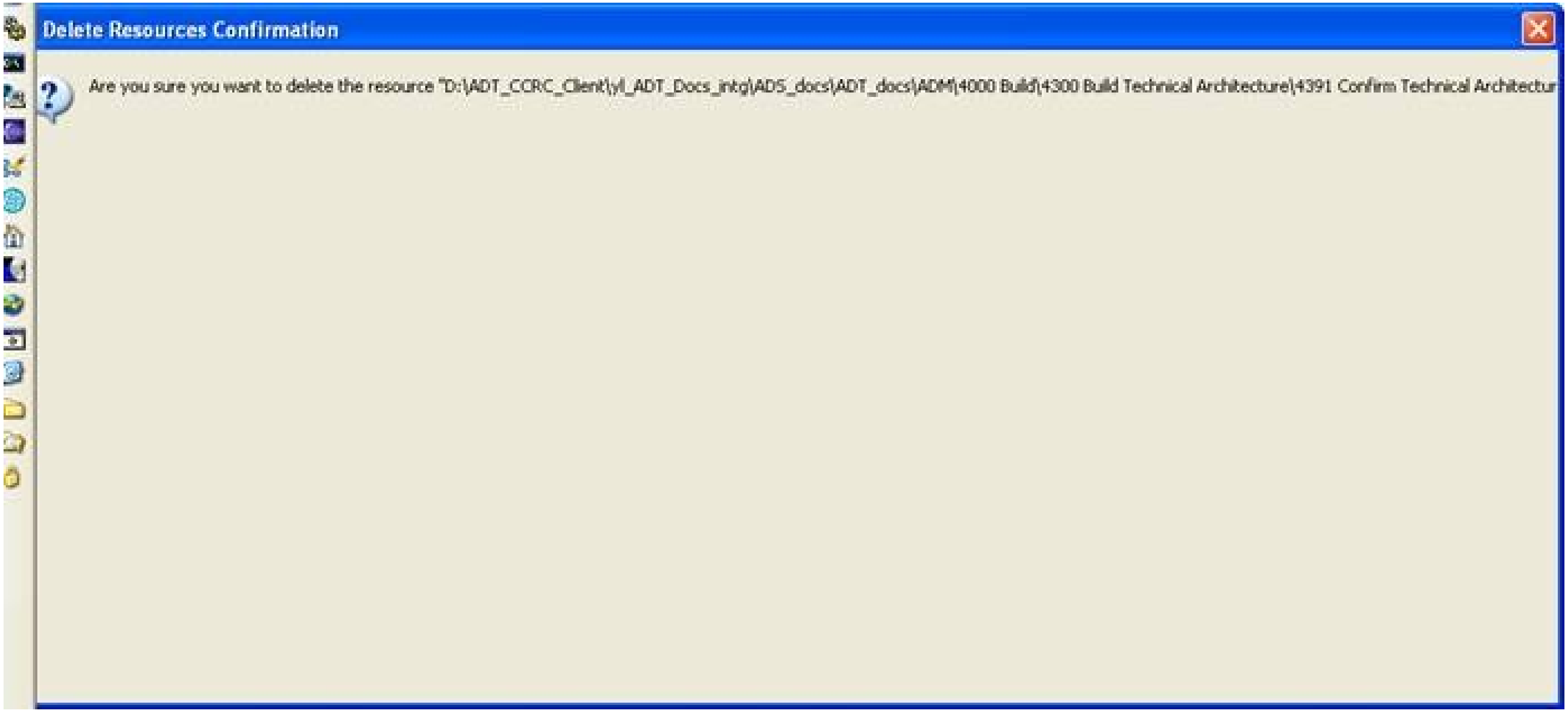
ICU Patients

Remember our ICU Monitoring problem? We had lots of references to patient there. How do we really establish identity?

- Name/Surname (risk homonymy)
- Bed number (risk loosing track upon moving)
- Patient number (risk re-assigning a new one in the future)
- Etc.

Identities h-concerns

- Can we *always* provide unique Ids?
- Even if we can, is that better for the user?

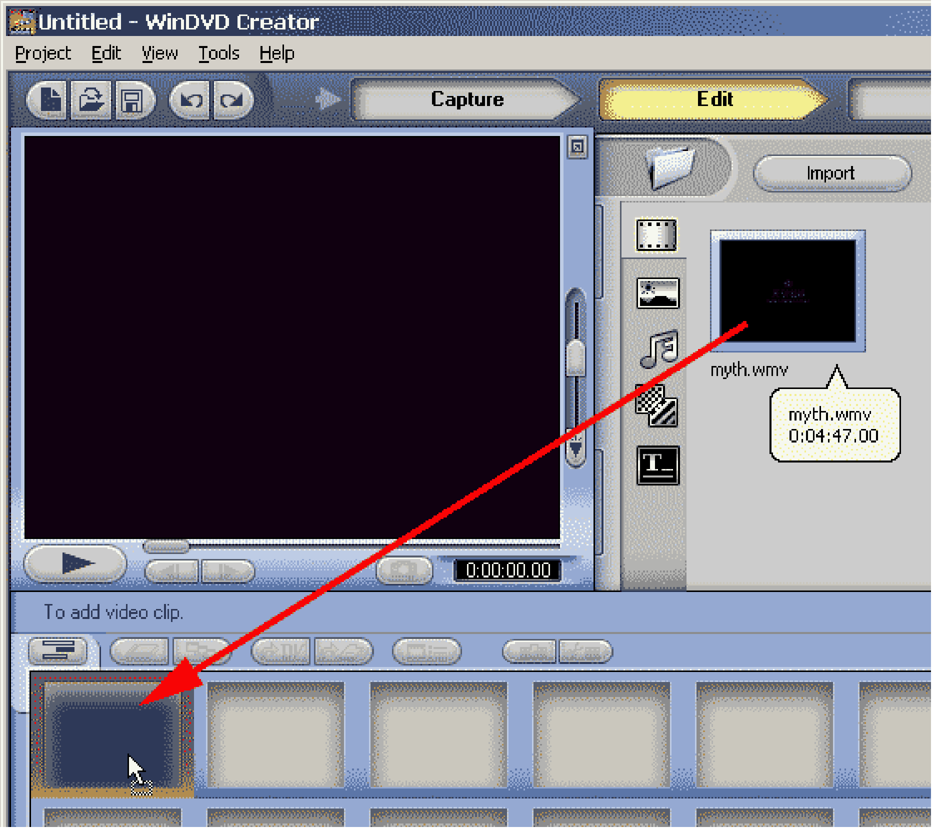


Delete Resources Confirmation

Are you sure you want to delete the resource "D:\ADT_CCRC_Client\yl_ADT_Docs_intg\ADS_docs\ADT_docs\ADM\4000 Build\4300 Build Technical Architecture\4391 Confirm Technical Architectur

Identities h-concerns

- Are icons or other forms of graphical representation enough to establish identity?



Completeness h-concerns

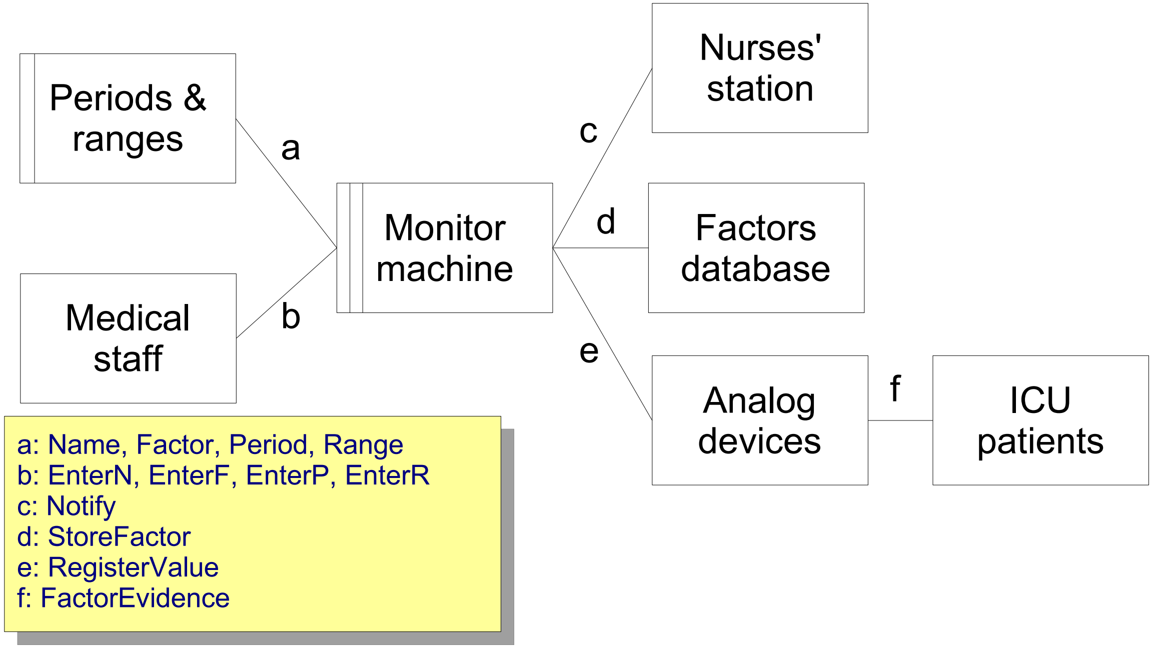
- We are confident that we have caught all relevant domains, phenomena, etc.
- Can “holes” in the user interface suggest more phenomena or new domains?
  - e.g., maybe the GUI has a “Cancel” button whose related event Cancel has not been considered in our modeling?
- Can standard UI practices be used to drive further elicitation?
  - The user did not ask for configuring the colours
  - Maybe we can propose it as a gizmo?

Other common D-concerns

- **Overrun**
  - One party of a communication too fast/slow for the other
- **Initialization**
  - Joining a system, self-configuration, discovery
- **Reliability**
  - Node or infrastructure fails
- **Identities**
  - How to define Globally Unique Identifiers
  - How to define proper scoper for non-GUID

Final exercise - 1

- Go back to the **ICU patient monitoring problem**
- Identify its sub-problems
- Fit them to problem frames
- Consider the concerns of each frame
- Prepare a specification for the Monitor machine

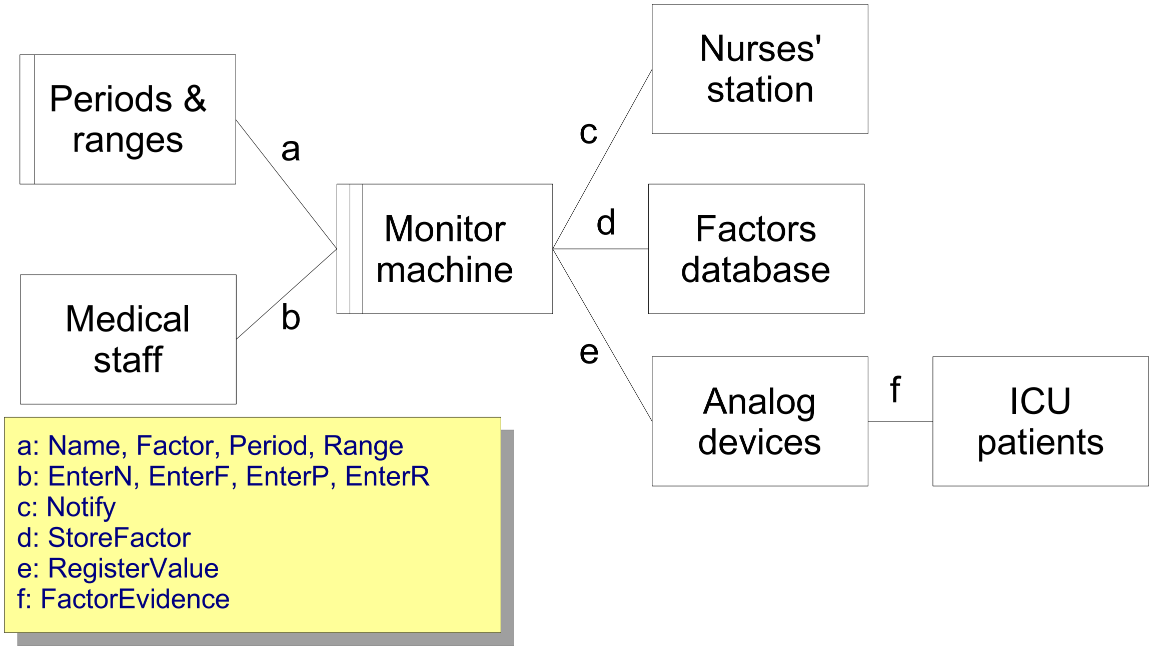


a: Name, Factor, Period, Range
b: EnterN, EnterF, EnterP, EnterR
c: Notify
d: StoreFactor
e: RegisterValue
f: FactorEvidence

- Put forward a tenable correctness argument for your specification
- Which implementation technology (hardware, OS, language) would you use for such a project?

Final exercise - 2

- Consider the h-concerns of each frame
- Also consider the generic h-concerns
- How would you realize a user interface for the Monitor Machine?
- List the things in the UI that make you feel uneasy



a: Name, Factor, Period, Range
b: EnterN, EnterF, EnterP, EnterR
c: Notify
d: StoreFactor
e: RegisterValue
f: FactorEvidence

- Sketch out use cases for the Monitor Machine, and prepare a storyboard of what the user interface would look like
- List 10 ways in which user behaviour can lead to utter failure regardless of your best efforts