Extending the Socio-economics of Software Architecture

Alistair Sutcliffe

University Lancaster &
University of Manchester

WICSA-ECSA Helsinki, August 2012

with thanks to
Sarah Thew (Manchester) and Pete Sawyer (Lancaster)

Presentation Aims

1. Argue for the importance of modelling Conceptual System Architecture

   - Requirements Engineering meets SE/Software Architecture

2. Convince you that ‘people issues – values’ have strong implications for software architecture design

   - Human Factors meets SE/Software Architecture

3. Map out a research agenda for extending the socio-economics of Software Architecture

Presentation Outline

- Part 1: Requirements Reuse and Conceptual System Architecture
  - Background
  - Problem description- healthcare application
  - Monitoring and Awareness system architectures
  - Adaptive system architectures

- Part II: Implications of User Values for System architecture
  - Value based Requirements Engineering
  - Values in system design

- Conclusions & research agenda

Part I

Reuse of

Conceptual System Architecture

Background

- Plenty of material on Software Architecture @ the Design level
  - from Garlan and Shaw onwards
  - Bass, Kazman et al (2003)
  - GOF patterns (Gamma et al 1994)
  - POSA series (Buschman, Schmidt et al 1996- 2008)

- But not so much on Architecture @ the Conceptual – Requirements level
  - Folwer (1997), Analysis Patterns
  - Service Oriented Patterns maybe ? IBM Web Service patterns, Oracle SOA patterns, http://www.soapatterns.org
  - Product Lines maybe ? Clemens & Northrop (2001), Pohl et al (2005)
  - Withall (2007), Software Requirements Patterns
  - Jackson (2000) Problem frames- more abstract
  - Sutcliffe (2002) Domain Theory- Object System Models

The problem: Mild Cognitive Impairment (MCI)- Alzheimer's disease

- Research Question- Can we detect early signs of MCI from peoples' use of computers and persuade them to have follow up diagnostic checks ?

- Approach- detect early signs of MCI from records of computer use- data and text mining. Give feedback to users and their doctors for follow up checks.

- Some problems

  - how accurate will diagnosis from computer user be ?

  - what is the danger of false positives ?

  - how can the system reassure the user and encourage follow up action ?

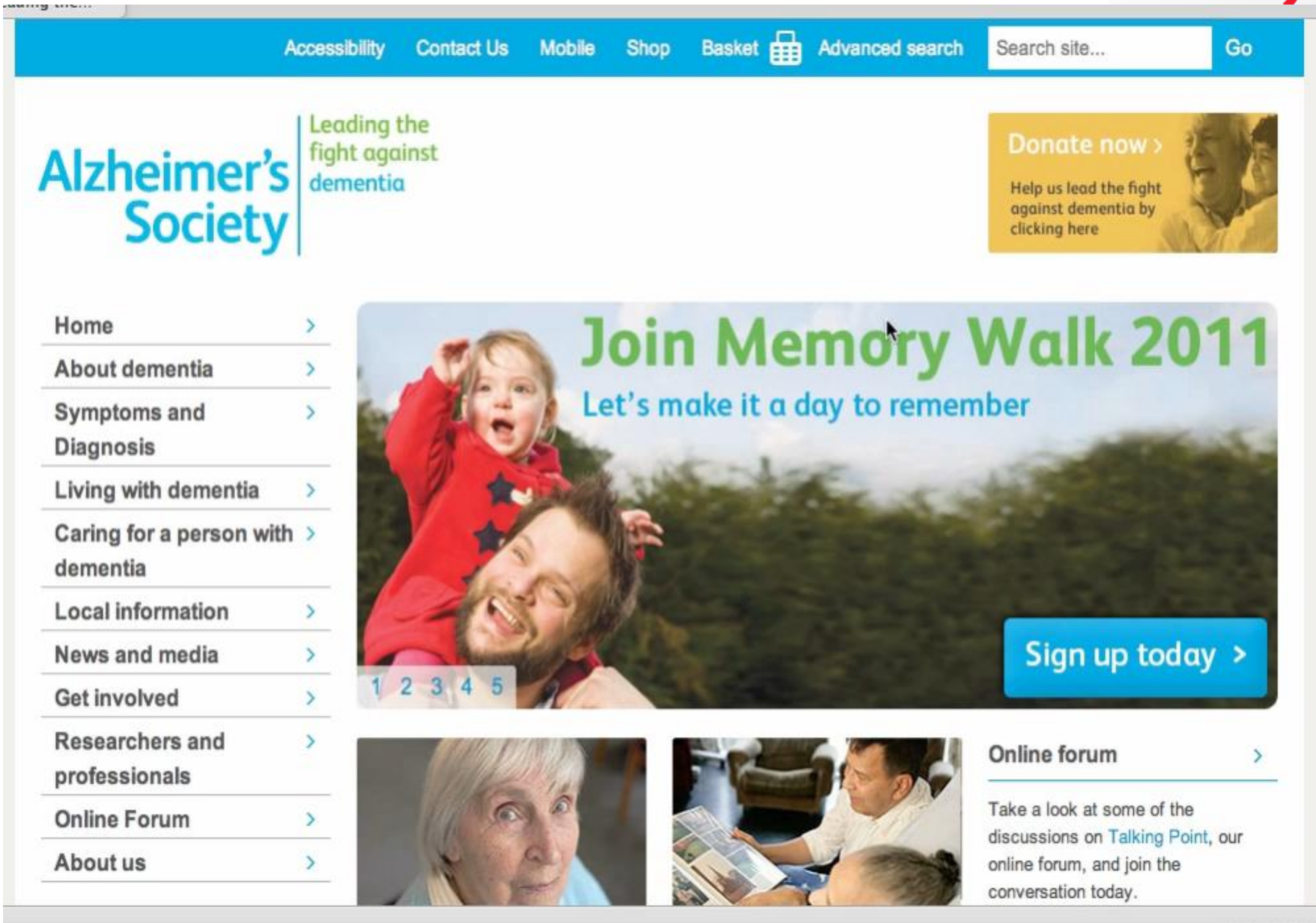
  - privacy, emotional issues, empathy, self efficacy.

MANCHESTER 1824

The University of Manchester

LANCASTER UNIVERSITY

In association with



Design Brief

(architecture requirements)

SAMS – Software Architecture for Mental Health Self-Management

- Solution needs to be as generic as possible

  - economic driver to address a wider class of analogous health care problems

- Distributed application- monitoring in users' homes, multi-platform installations

- Privacy and security (Data protection act, ethical issues)

  - client- server configuration, secure data transmission etc

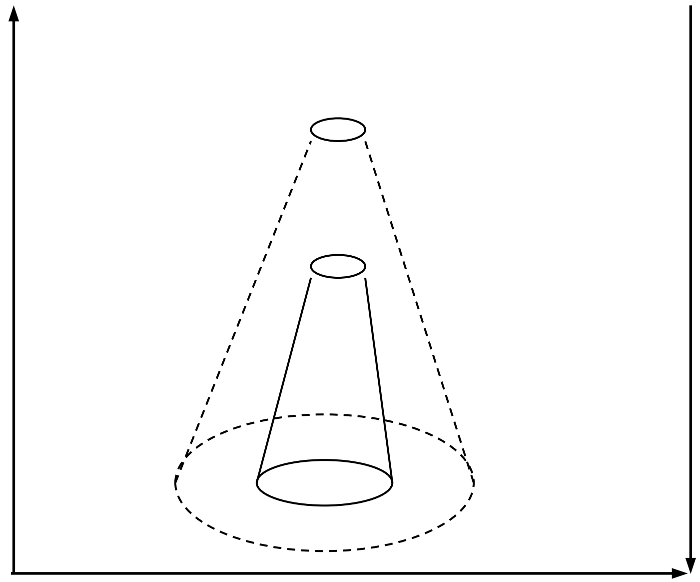
- Reduce development costs- software reuse

Identifying the Problem Class

- To produce a generic architecture we have to identify the range of 'analogous' applications

  - but how abstract should we aim to be ?

increasing abstraction



cost of specialisation

increasing detail and reuse utility

potential revenue: number of potential reuse targets

Problem Class
Self Aware, Adaptive Systems

- Awareness requirements (Mylopoulos, Souza et al 2011)

- Generic Monitors with adaptation ReqMon & EEAT (Robinson 2006, Fickas & Feather 1995)

- RELAX configurable adaptive systems (Sawyer, Whittle et al 2010)

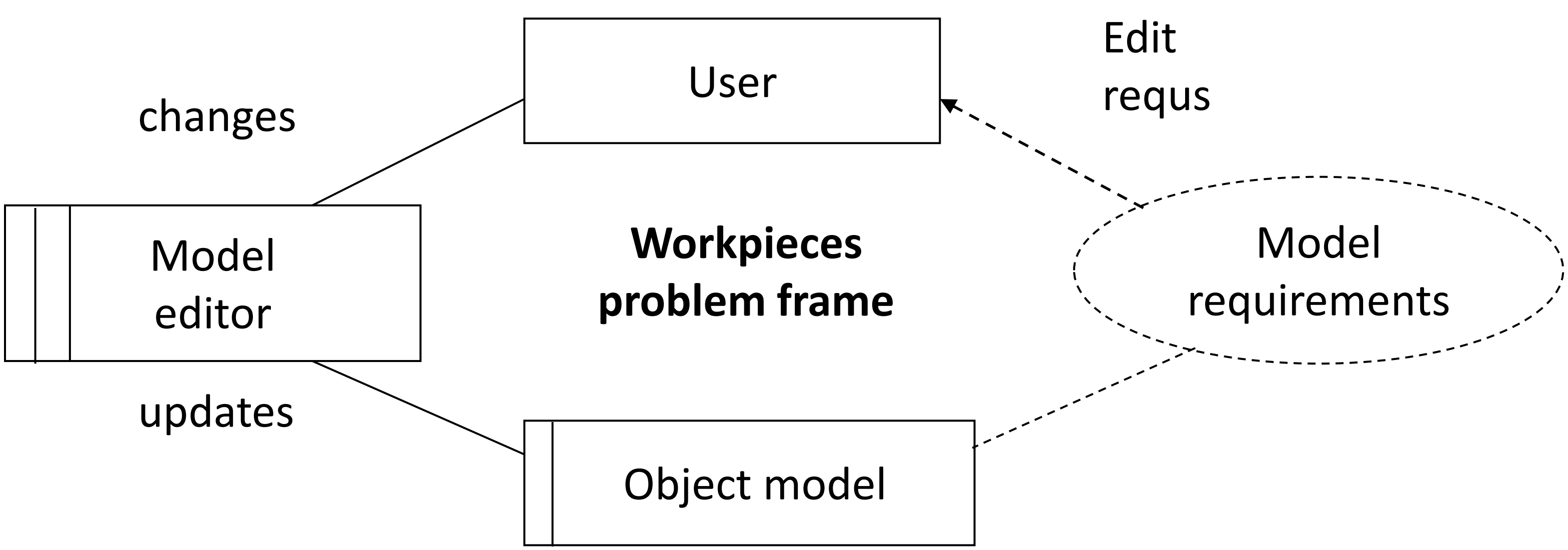
- Self aware systems (Ghezzi et al 2009)

- User Modelling –Adaptation in HCI, Recommender systems (Pu 2009, Dumais et al 2010)

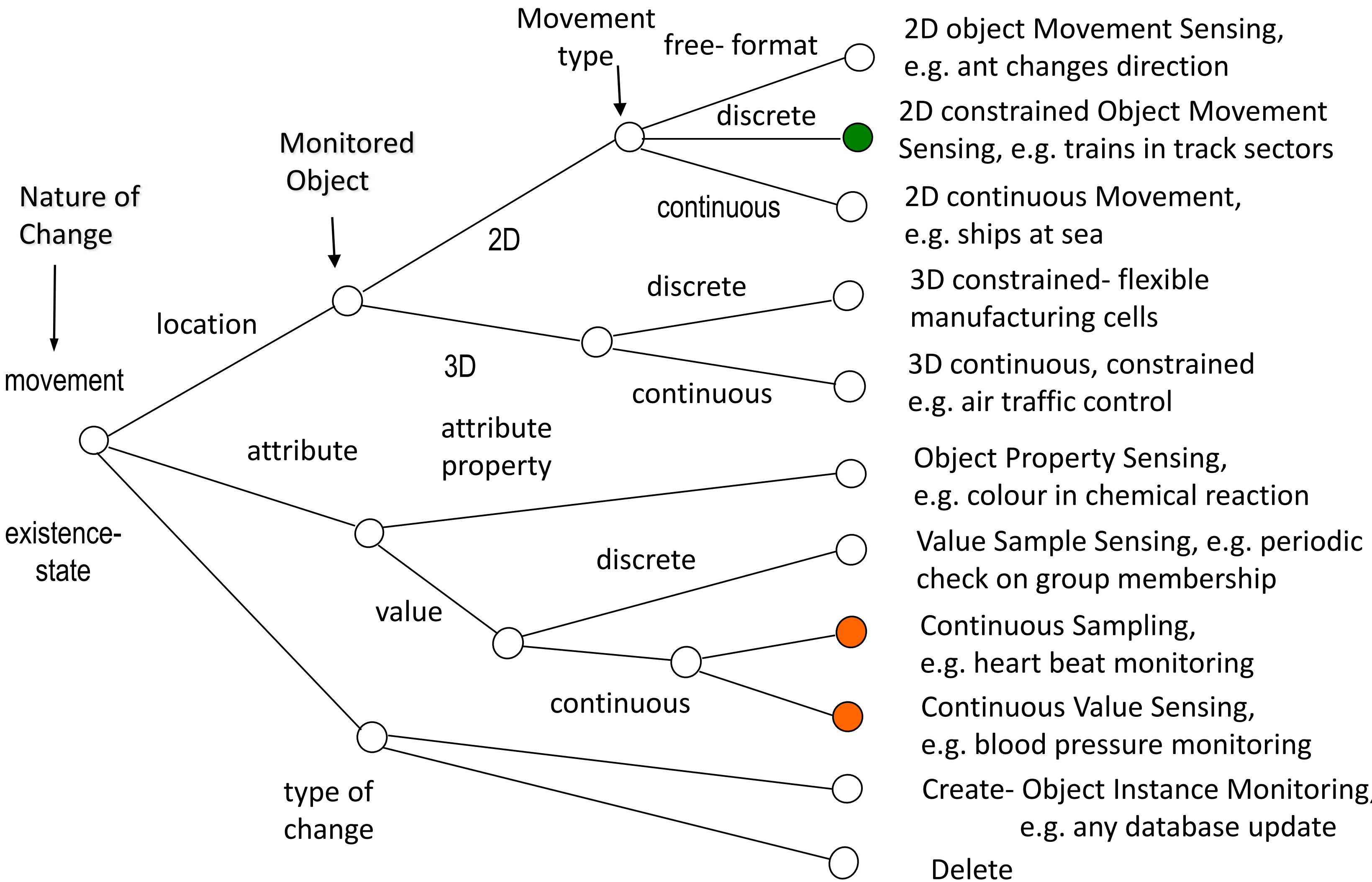
- Dynamic Planning in AI

Self Aware, Adaptive Systems

- A widespread class of problems, but ...
  - what defines this range of problems ?
  - are there any abstract models as starting points for {generic} architecture design ?
- Some models...but very abstract, no sub classes
  - in the solution domain GOF Observer pattern (Gamma et al 1994)
  - in the problem domain Jackson's problem frames (Jackson 2000)

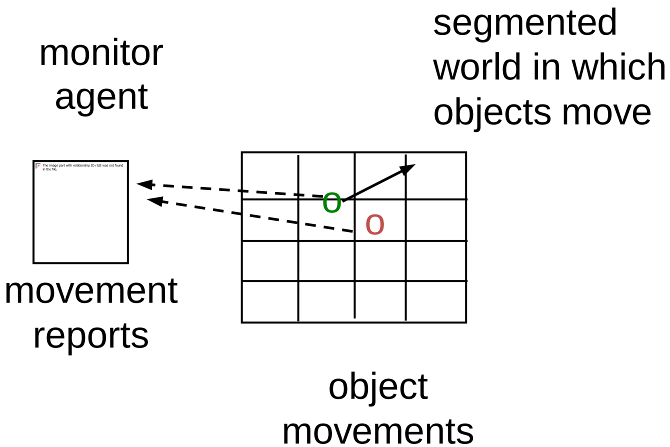


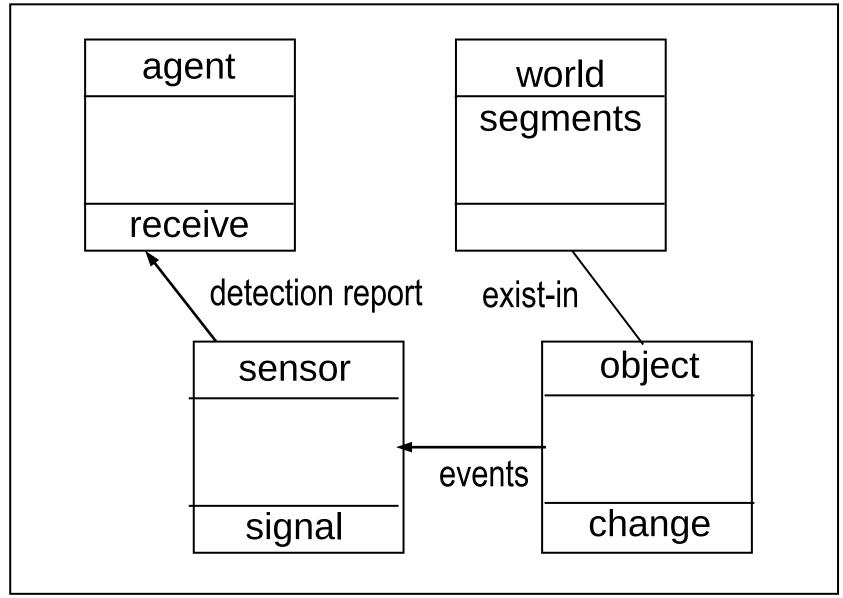
Self Aware, Adaptive Systems-
the Domain Theory view



Object Sensing System Models (monitoring, sense making)

Level-2 class Spatial Object Sensing





Generic Requirements (GR)

1. System model
2. Event filters
3. Event pattern monitor
4. Event interpretation
5. Trace history

Design Issues

1. Detectability of events
2. Fidelity of detection
3. Sampling frequency
4. Identifying events
5. Accurate interpretation

Awareness Requirements
(Souza, Mylopoulos et al 2011)

1. Event awareness

- Monitors for Single events (semaphores) and simple event patterns
    - detect exceptions and unexpected events
    - omissions, co-missions, early/late events (Hollnagel 1999)
    - patterns across multiple event streams
- Interpreters for more complex event patterns
    - match event patters to normal behaviour
    - detect exceptional patterns, alternative paths etc
    - interpret patterns in context (e,g, mobile awareness)

2. Performance- Conceptual awareness

- Data capture for event (and state/context) history

- Interpreters for complex patterns
    - model based interpretation
    - reasoning to infer higher order semantics (intent, concepts, trends, etc)
    - data and text mining, image/ audio recognition

- Understand the external world, adapt system to contextual changes

Monitor Types

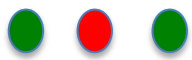
- Hard Monitors- Awareness requirements which can be captured automatically (or set as thresholds, targets, indicators, etc)
  - simple event analysers
  - compound event analysers- sequences, cumulative events
  - context analysers- event and states
  - complex event analysers, data miners with history

- Soft Monitors- Awareness requirements which can only by captured indirectly by people
  - by observation, interviews
  - surveys
  - standards compliance, certification
  - running tests, drills to check system performance
  - decision support analysis tools (e.g. statistical tests)

Hard (state/event) Awareness

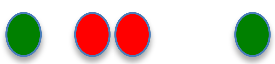
- State value, discrete, continuous, boolean

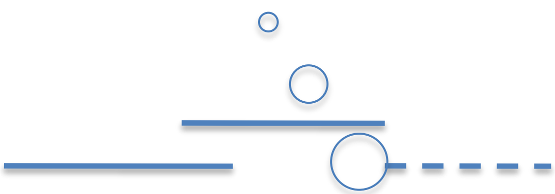
- Event identity



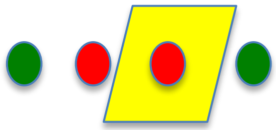
- Event patterns

- Temporal patterns





- Event –state monitors

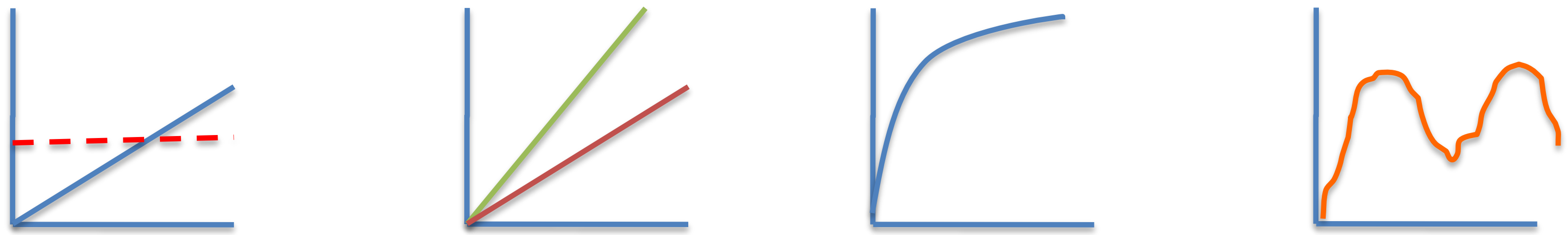


For an event pattern taxonomy
See Hollnagel (1999)
CREAM

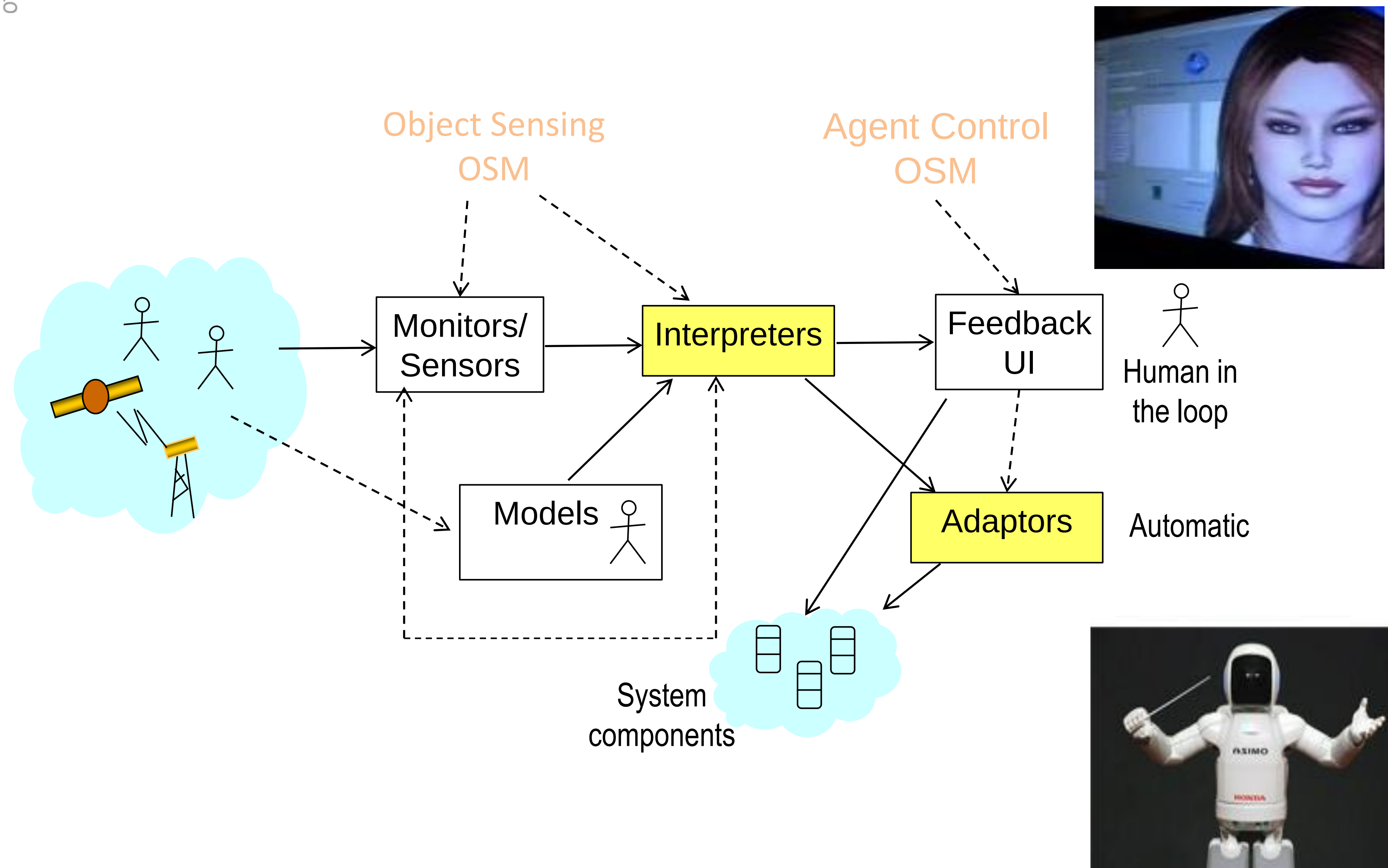
Performance awareness

- Aggregate data from event level monitors
  - over time
  - across individuals
  - classify events, categories, distributions
  - data miner, classifier components

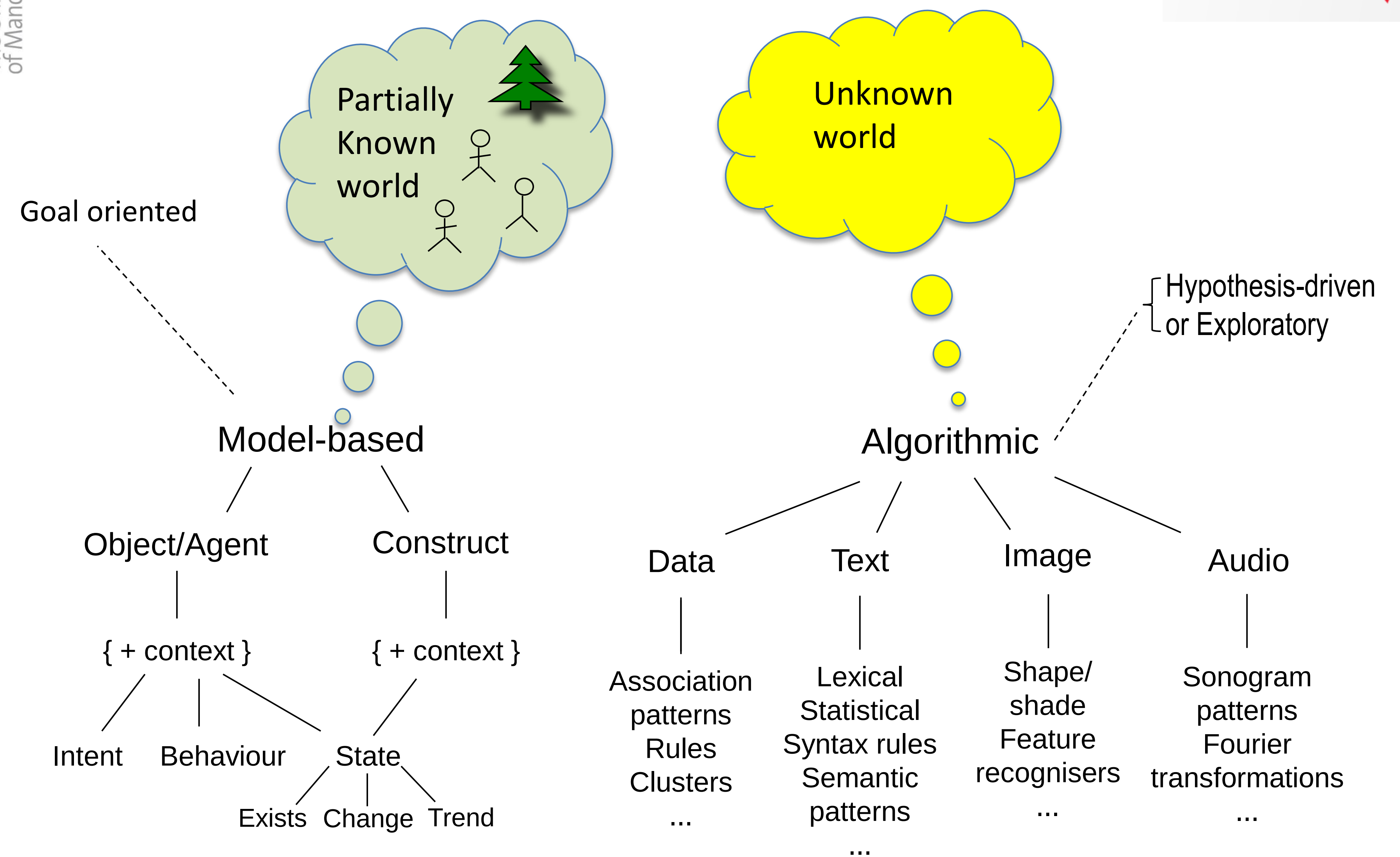
- Compare aggregated data against a target (threshold, indicator) or for desired patterns



Self Aware, Adaptive Systems Architecture



Interpreters



SAMS: Object sensing (People awareness)

- Agent (People) Monitors

  -monitoring values, states/ properties of agents,

  e.g. health care blood pressure, body temperature,

  cognitive states (memory, reaction time)

  - monitoring agent behaviour

  e.g. heart rate, respiratory rate, gestures, movement,

  analysing computer operation in email

  - monitoring intent and emotional state

  e.g. stress by heart rate and GSR,

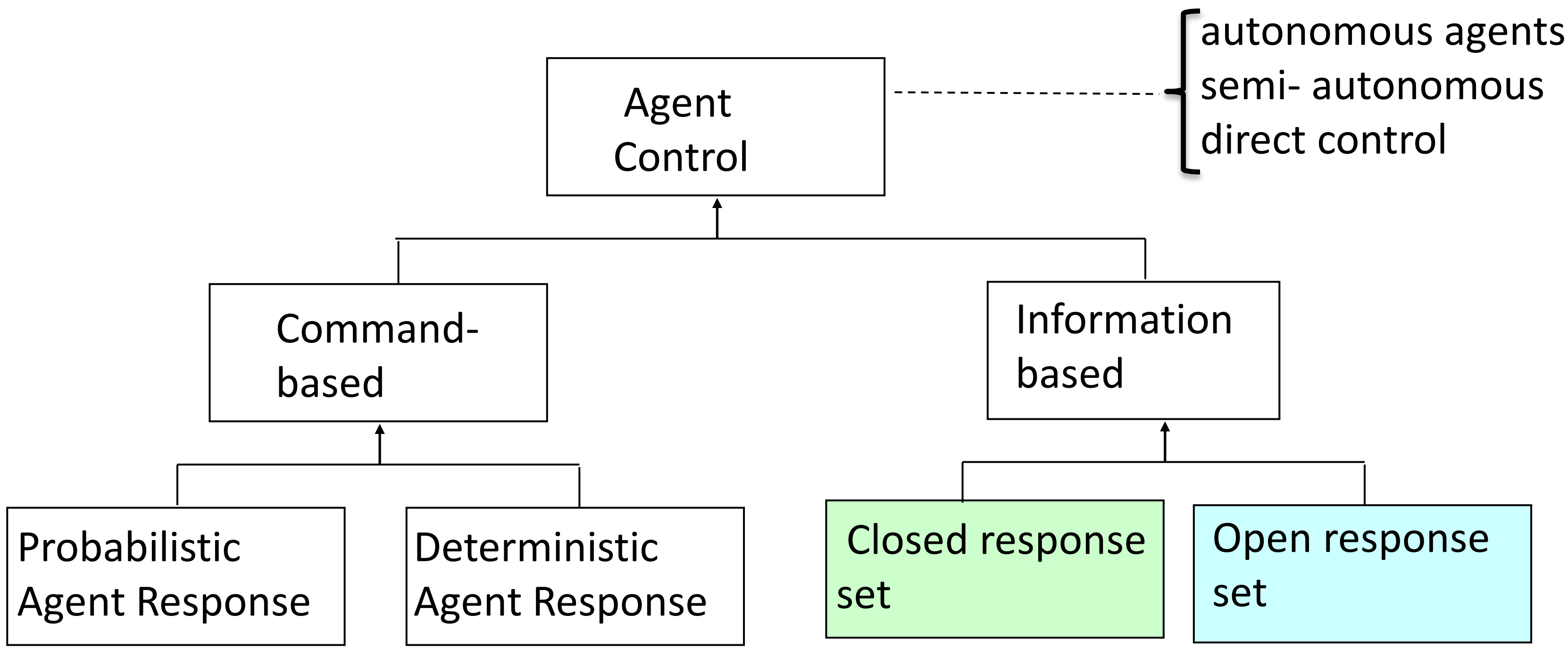
  intent from behaviour. affect from text

  -performance monitors

  e.g. exercise routines, calories burned, aerobic exercise level

  mental performance (MCI)

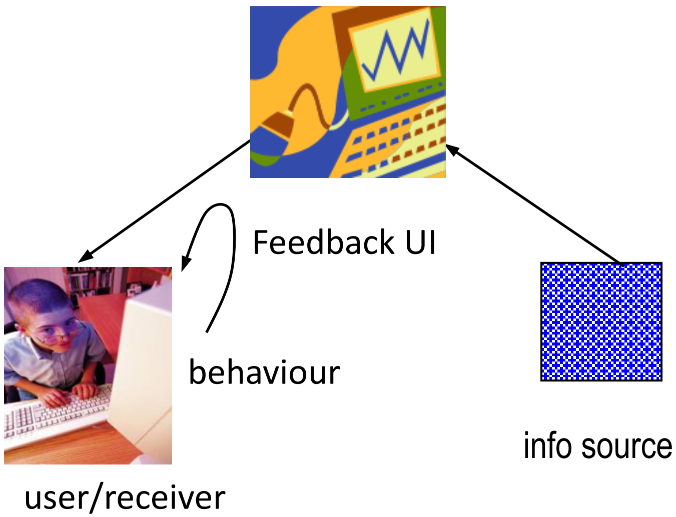
Agent Control OSM Family (adaptation component)

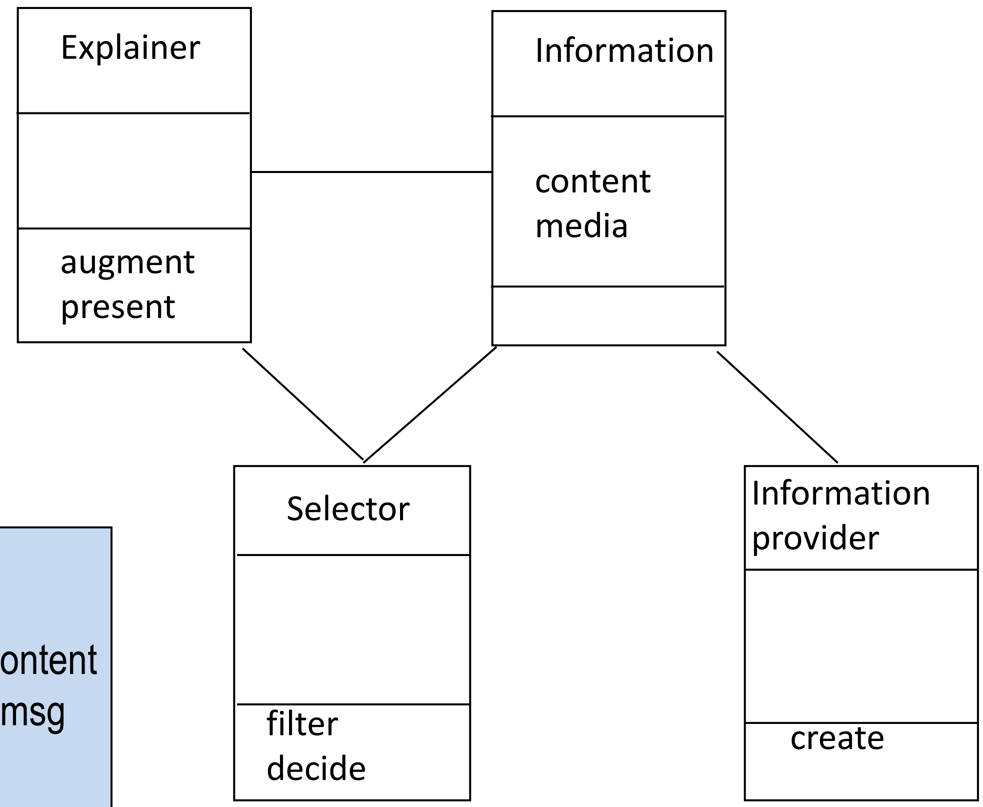


command & control systems
human / automated agents
close- loose coupling

human in the loop/ intelligent agents
explanation and persuasive systems
recommenders

Agent Information response- open





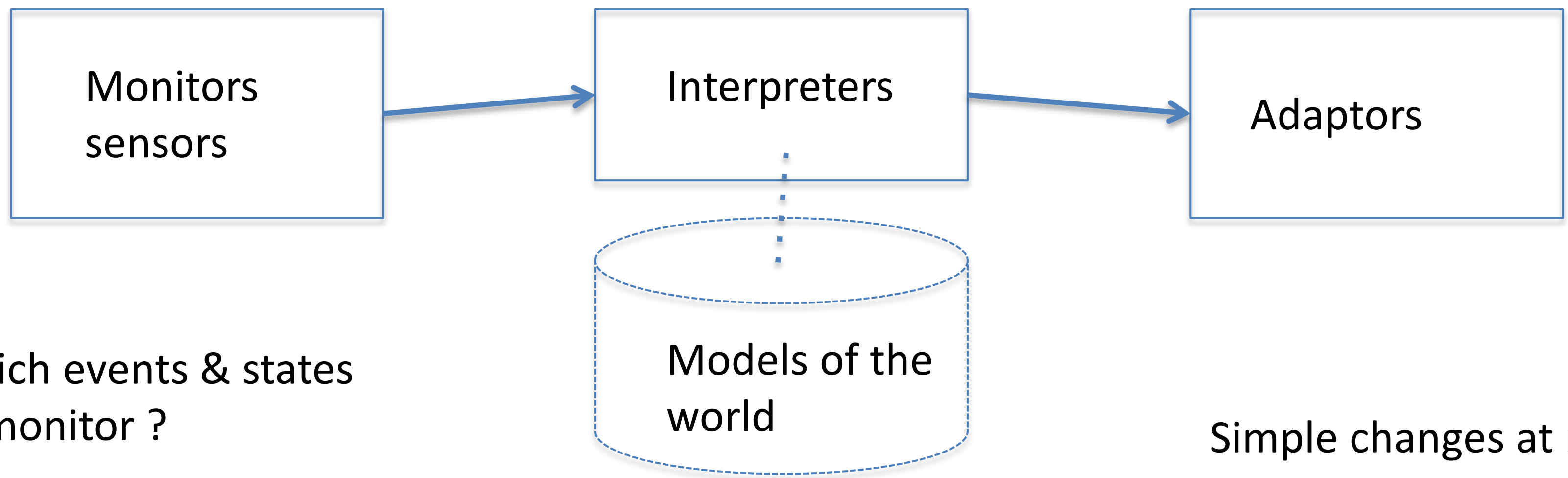
Generic Requirements

Information Presenters
Filters
Highlighters
Customisers
Interactive controls
Media

Design Issues

Selection of msg/content
Matching users to msg
Quantity of info
Delivery pace
Delivery-emotive effects
Argumentation

Object Sensing- Adapting Conceptual Model
@ the event level



Which events & states to monitor ?

Active or passive sensors ?

Event/state detectability

Fidelity of monitoring ? (time, signal type..)

Interpreting simple Events

Event patterns

Higher order states

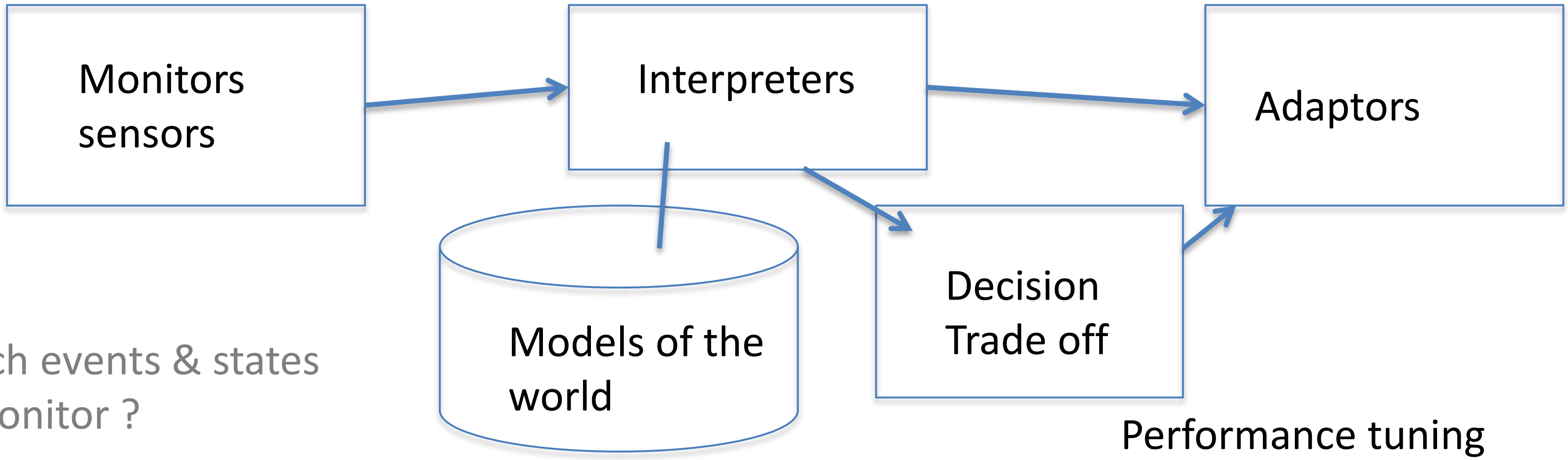
Simple changes at run Time

Response actions

Rule/method level changes

Delegation

Object Sensing- Adapting Conceptual Model @ the Performance level



Which events & states to monitor ?

Active or passive sensors ?

What fidelity of monitoring ? (time, signal type..)

How long (time period)

Scope (population, area, etc)

Interpreting Event patterns

Higher order constructs states, intent, models

Data & Text Mining Learning Algorithms

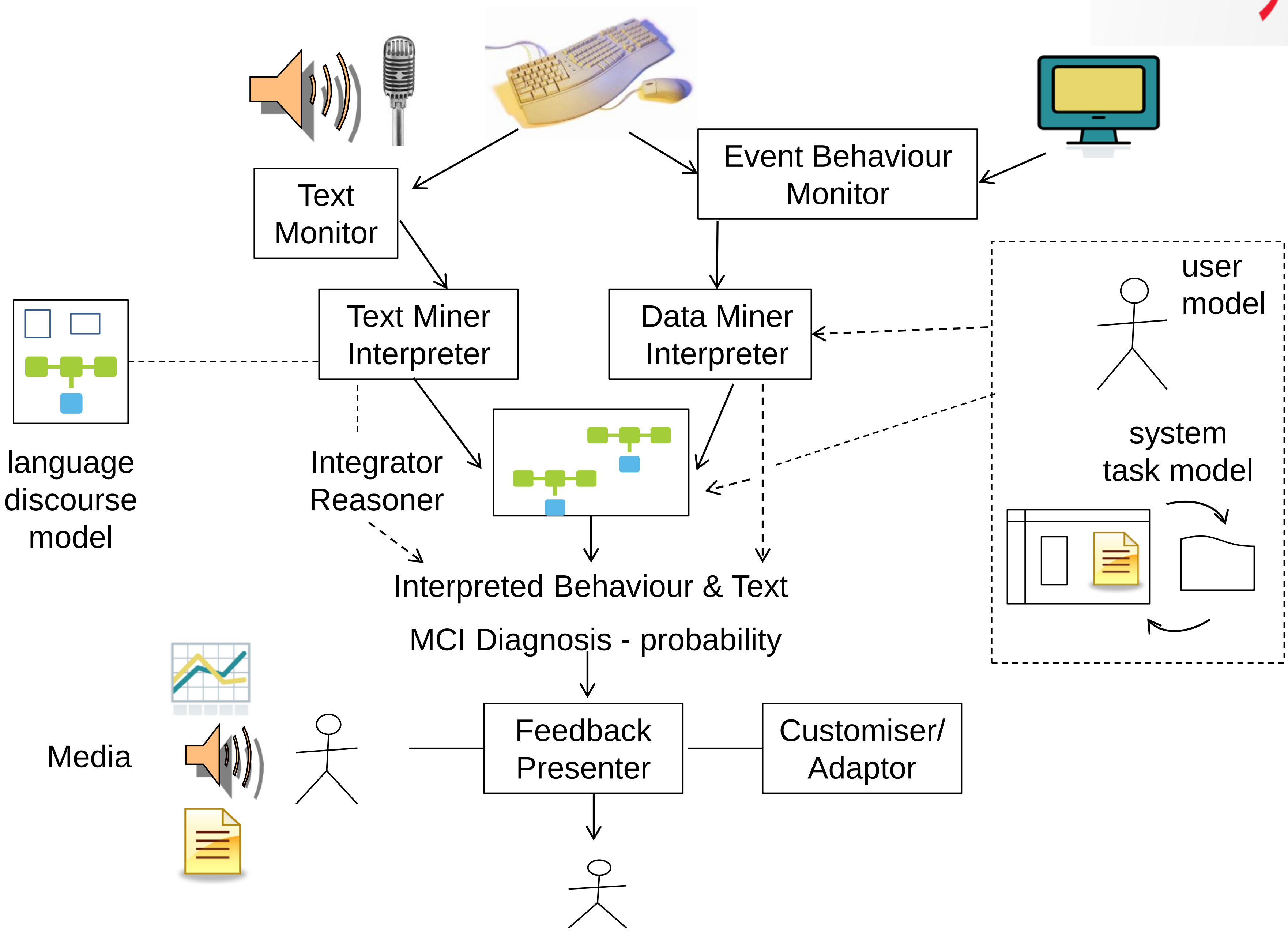
Performance tuning

Component selection

Delegation

Requirements change {new designs, Versions, product line Feature adaptation}

SAMS Conceptual Architecture



Knowledge (conceptual model) Reuse
SAMS Architecture

- Design and selection of performance monitor components- data miners (Open source libraries)
- Requirements and design of text miner components
- Selection of a mix of event and performance monitors (Open source)
- Choice of feedback UI- adaptation facilities
- System- architecture integration
- Ability to explain architecture- design options to users (medical researchers and participant volunteers)

Part II

Design implications of User Values

for System Architecture

Soft Issues, Values & Architecture

- Values- stakeholder beliefs, attitudes, opinions

- Surely this is all in the social part of systems....

- But people are in the loop of most systems...

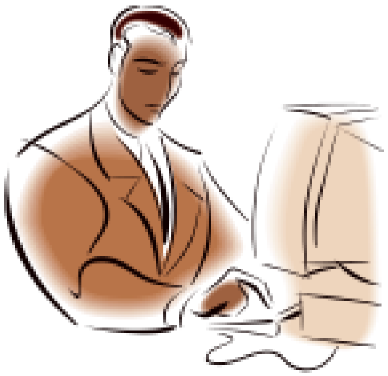
- Self aware- Adaptive systems are widespread

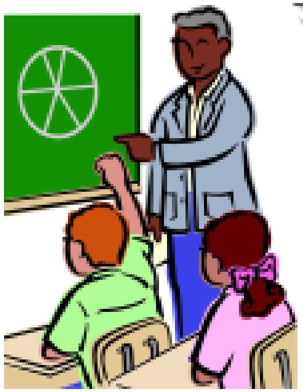
  - in healthcare, patient monitoring

  - in ecommerce, recommender systems

  - in education, training systems

  .... and many other domains





So what are ‘Values’ ?

Motivations
goals

influenc
e

Decisions
Actions

influence

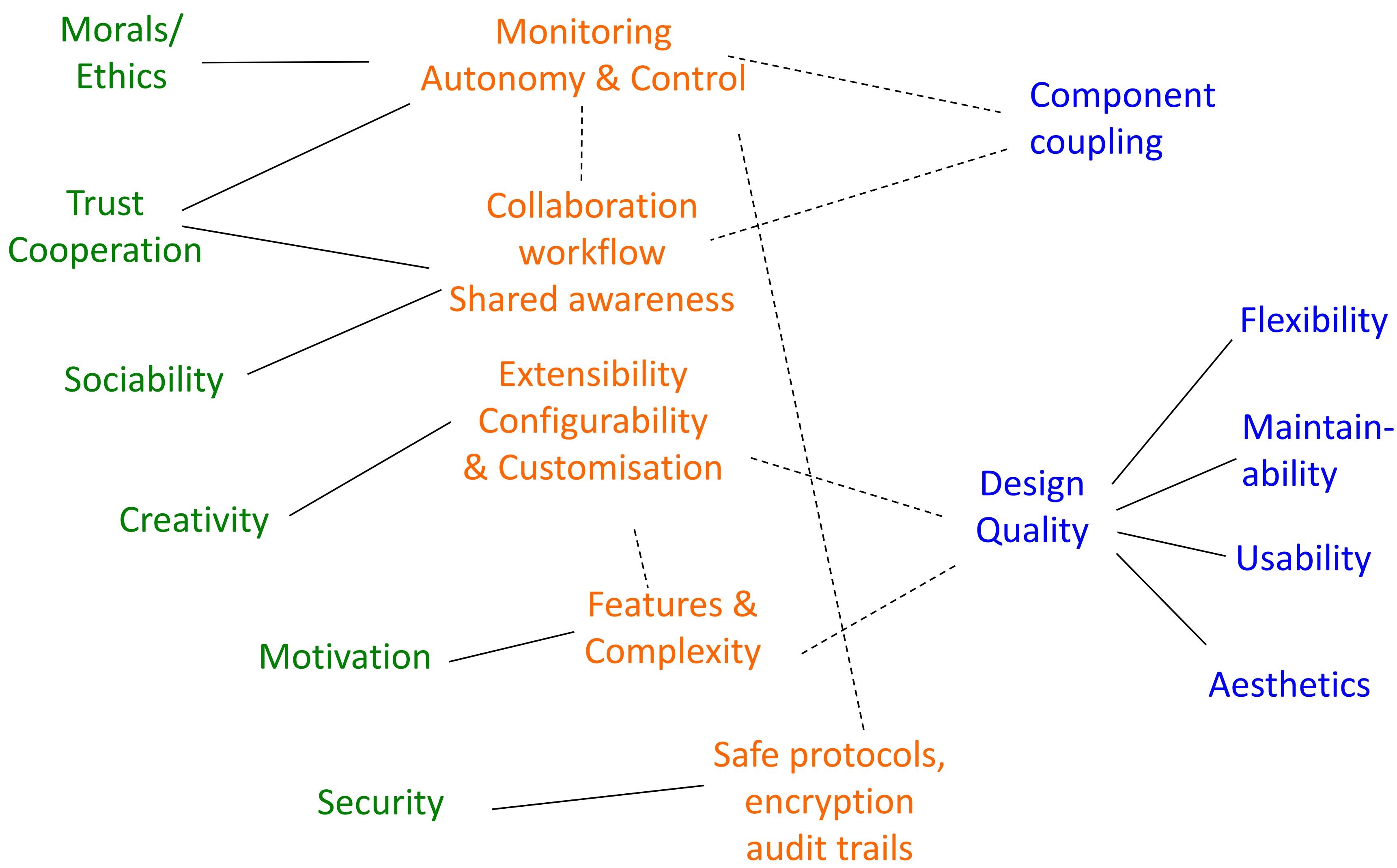
Emotions
Feelings

Values
Beliefs
Attitudes

Personality

- Related to non functional requirements- e.g. security, privacy, usability,
- Users’ beliefs, attitudes, concepts, some are generic, other transient-cultural, e.g. green-environmental values
- Value sensitive design – Freidman et al - www.vsdesign.org

Values- Architecture implications



Value based Requirements Engineering

(Thew & Sutcliffe 2008)

- Guidance about ways to identify values, motivations and emotions, & potential project impact
- Informed by analyst interviews, project reports & psychological theory.

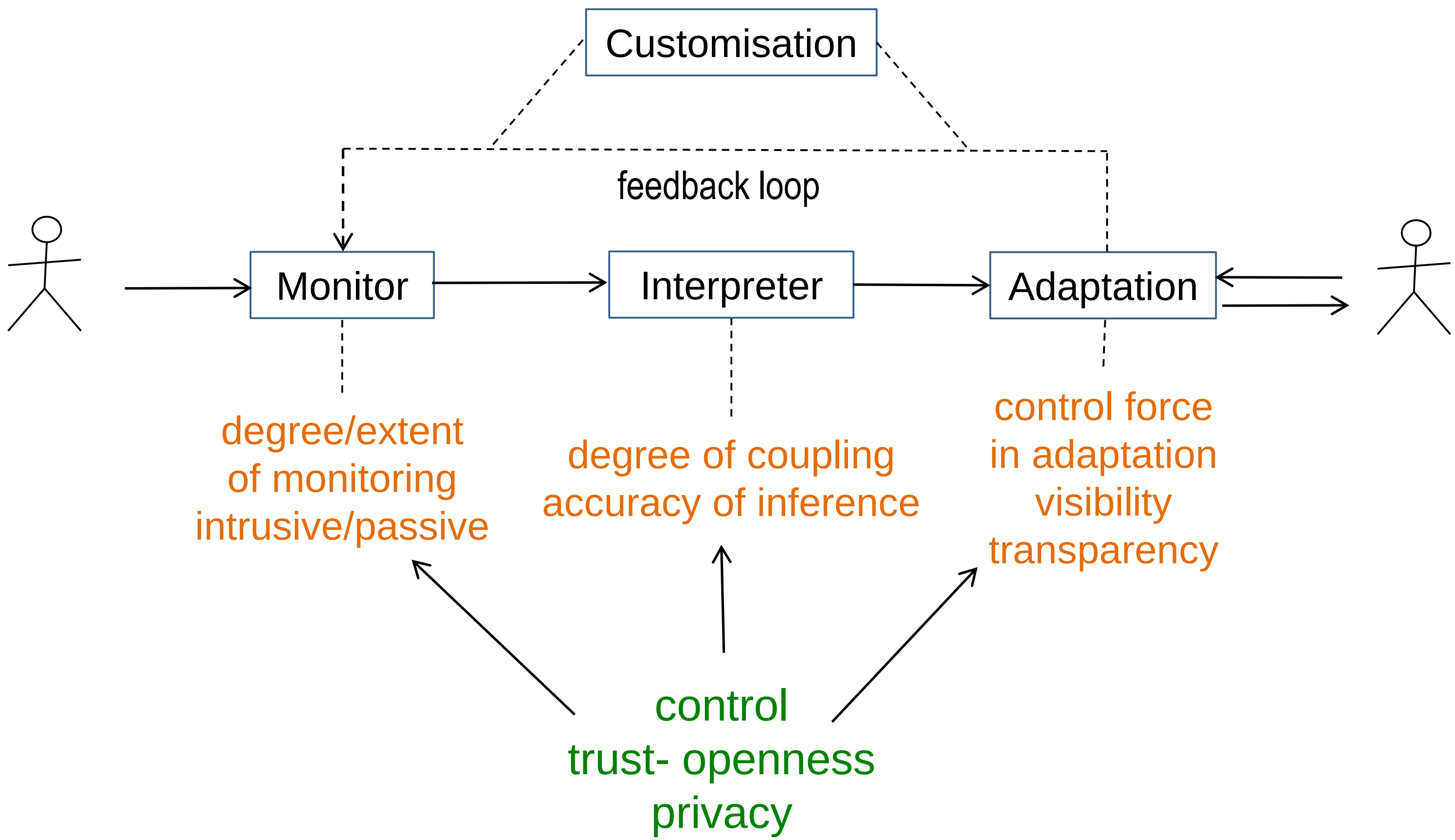
Value concept	**Related terms**	**Potential sources**	**Process implications**
Trust	openness integrity loyalty responsibility reliability	Relationships with other individuals /departments Privacy policies	Less control milestone checks improved team confidence
Collaboration	cooperation friendship sympathy altruism	Relationships with others Relationships: awareness of others – office politics	Improved team cooperation shared awareness

MANCHESTER 1824

The University of Manchester

LANCASTER UNIVERSITY

Impact of Values



Values- impact on SAMS

- Trust and privacy concerns, user control over data and system, visibility and explanation facilities.
- User control- configuration and customisation of architecture- more/less analysis, extent of monitoring (e.g. +/- email content)
- Loose coupling between system components (Interpreters→ Adapters) users in the loop
- Accuracy and emotional sensitivities- Feedback UI design for communicating results (false positives problem)

Reflections-
Reuse & Conceptual System Models

- Room for conceptual models in reuse ?
  - ERPs commercially established... but address established business needs
  - Product lines, also established... but tend focus on engineering sector applications
  - Open source components vast choice, selection and composition problems
- Models and taxonomies for indexing software component libraries- link between problem and solution models to software components
- Knowledge reuse – integrating requirements engineering and software design

Reflections-
User Values and System Architecture

- Socio-Technical systems ‘thinking’ in design of software architecture
- Values link requirements – (user perspective) to software engineering- (design perspective)- see also Twin Peaks model (Nuseibeh 2006)
- Simple set of concepts and heuristics/ guidelines for architecture design
- Values critical for human in the loop systems- link Human Factors/ Human computer interaction to software engineering
- Values already present in Agile method Process (Beck 1999), need to add design implications

Research Agenda
Conceptual Modelling & Reuse

- Develop taxonomy of conceptual system models
- Apply conceptual models in practice – development methods are more than just process- knowledge reuse needs to be integrated
  - pattern books of models for RUP- UML ?
- Support tools for Reuse (model) Oriented Software Engineering- intelligent hypertext, design advisors
- Abstraction theory- a really difficult research challenge
  - so what is the ideal cut on abstraction ?
  - where are the optimal boundaries, granularity ?

Research Agenda

Socio-Economics of System Architecture

- Analysis methods, heuristics and patterns connecting human 'social issues' to software engineering and systems architecture

  - more than just values,

  .....emotional effects in interactive agents

  ..... social media architectures

  ...... robot architectures

- Values in the development process- tools for thought in agile methods

- Socio-economics of software architecture- costs- benefit analysis for system design

Conclusions

- I hope I have convinced you of the merits of conceptual modelling
- And the need for a Theory of Abstraction for system architecture
- The value of Values and how human issues should be incorporated system design
- And that requirements and software architecture need to work more closely together

"The inevitable intertwining of requirements and architecture design"

after Bob Balzar

Thank you

and any questions ?

Selected References

Endrei M, et al (2004). Patterns: Service- Oriented Architecture and Web Services, IBM/Redbooks.

Hollnagel, E. (1998). *Cognitive Reliability and Error Analysis Method: CREAM*. Elsevier, Oxford.

M. Jackson. (2001) Problem Frames: Analysing and Structuring Software Development Problems. Harlow: Pearson Education,

Withall S. (2007), Software Requirement Patterns, Wiley/Microsoft

Pohl,K., Böckle, G., van der Linden, F. (2005), Software Product Line Engineering: Foundations, Principles, and Techniques. Springer, Berlin

Clements, P. & Northrop, L. (2001), Software Product Lines: Practices and Patterns, Addison-Wesley Professional.

Bass, L., Clements, P., & Kazman, R. (2003), Software Architecture in Practice, Addison-Wesley.

Nuseibeh B., (2006), Weaving together requirements and architecture. IEEE Software 34(4), 115-117

Sawyer, P., Bencomo, N., Whittle, J., Letier, E & Finkelstein, A. (2010), "Requirements-Aware Systems A research agenda for RE for self-adaptive systems". in Proceedings, 18th IEEE International Conference on Requirements Engineering (RE '10), Sydney, Australia. Los Alamitos CA: IEEE Computer Society Press, 2010, pp. 95-103.

Whittle, J., Sawyer, P., Bencomo, N., Cheng, B., Bruel, J-M. (2010), "RELAX: A Language to Address Uncertainty in Self-Adaptive Systems Requirements", Requirements Engineering Journal. 15 (2), 2010. pp 177-196.

Souza, V.E., Lapouchnian, A., Robinson, W.S., & Mylopoulos, J. (2011) Awareness requirements for adaptive systems. in Proceedings of SEAMS '11 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pp 60-69, ACM Press

Sutcliffe, A. G. (2008). The socio-economics of software architecture. *Automated Software Engineering*, *15*, 343-363.

Thew, S., & Sutcliffe, A. G. (2008). Investigating the role of soft issues in the RE process. In *Proceedings of 16th IEEE International Requirements Engineering Conference RE 2008.* Los Alamitos CA: IEEE Computer Society Press, pp 63-66

Sutcliffe. A.G. (2009), "On the inevitable intertwining of requirements and architecture", in K. Lyytinen, P. Loucopoulos, J. Mylopoulos and B. Robinson (Eds). Design Requirements Engineering: A Multi-Disciplinary Perspective for the Next Decade. Berlin: Springer, pp. 168-85.

Sutcliffe, A. G., Papamargaritis, G., & Zhao, L. (2006). Comparing requirements analysis methods for developing reusable component libraries. *Journal of Systems and Software*, *79*(2), 273-289.

Papamargaritis, G., & Sutcliffe, A. G. (2004). Applying the Domain Theory to design for reuse. *BT Technology Journal*, *22*(2), 104-115.

Sutcliffe, A. G. (2002). *The Domain Theory: Patterns for knowledge and software reuse.* Lawrence Erlbaum Associates, Mahwah NJ.