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
  - 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
  - Withall (2007), Software Requirements Patterns
  - Jackson (2000) Problem frames- more abstract
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.
In association with Alzheimer's Society
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?

\[ \text{increasing abstraction} \quad \text{cost of specialisation} \]

\[ \text{increasing detail and reuse utility} \]

\[ \text{potential revenue: number of potential reuse targets} \]
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

Nature of Change

2D object Movement Sensing,
e.g. ant changes direction
2D constrained Object Movement Sensing,
e.g. trains in track sectors
2D continuous Movement,
e.g. ships at sea
3D constrained- flexible manufacturing cells
3D continuous, constrained
e.g. air traffic control

Object Property Sensing,
e.g. colour in chemical reaction
Value Sample Sensing,
e.g. periodic check on group membership
Continuous Sampling,
e.g. heart beat monitoring
Continuous Value Sensing,
e.g. blood pressure monitoring
Create- Object Instance Monitoring,
e.g. any database update
Delete

Movement type

Movement

location

existence-state

Monitored Object

attribute

attribute property

value

type of change

free-format

discrete

continuous

discrete

continuous

continuous
Object Sensing System Models (monitoring, sense making)

Level-2 class Spatial Object Sensing

- Monitor agent
- Segmented world in which objects move
- Movement reports
- Object movements

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 patterns 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 be 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

Object Sensing OSM

Monitors/Sensors → Interpreters

Models

System components

Agent Control OSM

Feedback UI

Adaptors

Human in the loop

Automatic

System components
Interpreters

Goal oriented

Model-based

Object/Agent

Construct

{ + context }

Intent

Behaviour

State

Exists

Change

Trend

Partially Known world

Unknown world

Algorithmic

Data

Text

Image

Audio

Hypothesis-driven or Exploratory

Association patterns

Rules

Clusters

... Statistical Syntax rules Semantic patterns

Shape/shade Feature recognisers

Sonogram patterns

Fourier transformations

Lexical

...
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)

Agent Control

Command-based

Probabilistic Agent Response
Deterministic Agent Response

Information based

Closed response set
Open response set

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

**Feedback UI**
- behaviour

**Info source**

**Selector**
- filter
- decide

**Information**
- content
- media

**Information provider**
- create
Object Sensing - Adapting Conceptual Model
@ the event level

- Monitors
  - sensors

- Interpreters
  - Models of the world
    - Models of the world

- Adaptors
  - Simple changes at run Time
  - Response actions
  - Rule/method level changes
  - Delegation

- 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
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

Text Monitor

Text Miner Interpreter

Event Behaviour Monitor

Data Miner Interpreter

Integrator Reasoner

Interpreted Behaviour & Text

MCI Diagnosis - probability

Feedback Presenter

Customiser/Adaptor

language discourse model

Media

system task model

user model
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’?

- 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

Motivations
  - goals

Personality

Decisions
  - Actions

Values
  - Beliefs
  - Attitudes

Emotions
  - Feelings

Influence
Values - Architecture implications

Morals/Ethics

Trust

Cooperation

Sociability

Creativity

Motivation

Security

Monitoring

Autonomy & Control

Collaboration workflow

Shared awareness

Extensibility

Configurability & Customisation

Features & Complexity

Safe protocols, encryption audit trails

Component coupling

Flexibility

Maintainability

Usability

Aesthetics

Design Quality

Security

Motivation

Creativity

Sociability

Cooperation

Trust

Morals/Ethics
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.

<table>
<thead>
<tr>
<th>Value concept</th>
<th>Related terms</th>
<th>Potential sources</th>
<th>Process implications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trust</td>
<td>openness</td>
<td>Relationships with other individuals /departments Privacy policies</td>
<td>Less control milestone checks improved team confidence</td>
</tr>
<tr>
<td></td>
<td>integrity</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>loyalty</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>responsibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>reliability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Collaboration</td>
<td>cooperation</td>
<td>Relationships with others</td>
<td>Improved team cooperation shared awareness</td>
</tr>
<tr>
<td></td>
<td>friendship</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sympathy</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>altruism</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Impact of Values

- Customisation
  - degree/extent of monitoring intrusive/passive
  - degree of coupling accuracy of inference
  - control force in adaptation visibility transparency

- Monitor
  - feedback loop

- Interpreter

- Adaptation
  - control trust openness privacy

- Feedback loop

The diagram illustrates the impact of values on various aspects of a system, including monitoring, coupling, inference, and adaptation, with feedback loops that maintain control and trust.
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

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