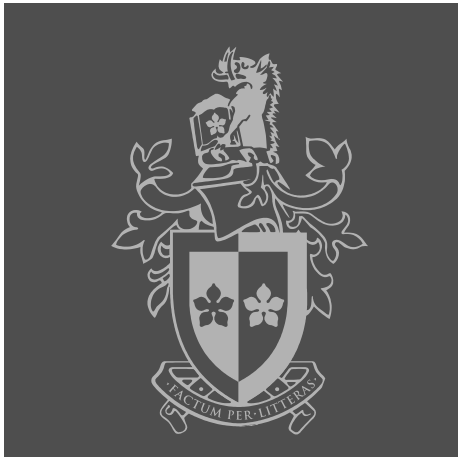
SWINBURNE UNIVERSITY OF TECHNOLOGY

Agent-Oriented Software Engineering

Introduction into the Paradigm



Complex Software and Software Paradigms

- Construction of software systems is one of the most complex endeavours ever undertaken by humans.
- Wide range of engineering paradigms to cope with complexity that claim to
  - Ease up the engineering process, or
  - Extend opportunities to successfully model complexity.
- Quantitative data for testifying claims is simply non existent for no single paradigm.
- So it is with Agent-Oriented paradigm
  - It is no silver-bullet,
  - Increasing number of deployed applications underpin that the agent-oriented paradigm is well suited for the construction of complex distributed systems.

SWINBURNE UNIVERSITY OF TECHNOLOGY

Key Concepts (1) - Agent

An Agent is an encapsulated computer system situated in some environment and capable of flexible, autonomous actions in that environment in order to meet its design objectives.

- Opinions about definition strongly vary
- Generally agreed main points
  - Agents are clearly identifiable problem-solving entities with well-defined boundaries and interfaces, situated in a particular environment over which they have partial control and observability
  - Agents receive inputs related to the systems state through sensors and they act through effectors to change environment because they are designed to achieve goals
  - Agents have some objectives to achieve autonomously
  - Agents have control over their internal state and their behaviour to exhibit proper behaviour to solve objectives
  - Agents are reactive (to changes in their environment) and proactive (to act in order to achieve objectives)

Key Concepts (2) - Interaction

- System modelling with Multiple agents
  - because of decentralised nature of the problem
  - multiple loci of control
  - multiple perspectives or computing interests.
- Agents interact on the knowledge level (high-level ACL based on Speech Act)
  - To achieve individual or collective goals
  - Rich social interactions such as cooperation, coordination, negotiation (in contrast objects interact on a pure syntactic level)
  - Flexible communication to handle just partial knowledge/control about/over environment
  - Handle changes that were not foreseen at design time.
- Central question
  - Which goal to follow and when and how to interact with whom to solve a problem?

Key Concepts (3) - Organisations

- The interplay between the sub-systems and between their constituent components is most naturally viewed in terms of high-level social interactions
- Problem space implies an organisational context
  - Giving roles and so different objectives to different agents,
  - Roles can change and so the relationships between different agents,
  - Relationships can be highly dynamic.
- High impact of organisational relationships and structures on system behaviour
  - Explicit structures and flexible mechanisms are central to the agent paradigm (interaction protocols, roles, goals).

SWINBURNE UNIVERSITY OF TECHNOLOGY

Agent-Oriented Decomposition (1)

- Complex systems consist of a number of related sub-systems
- Sub-systems work together to achieve the functionality of their parent system
- Within a sub-system, the constituent components work together to deliver the overall functionality

- *Each component can be thought of as achieving one or more objectives.*

(decompositions based upon functions/actions/processes are more intuitive and easier to produce than those based upon data/objects is even acknowledged within the object-oriented community (e.g. see, B. Meyer (1988) "Object-oriented software construction" Prentice Hall.)

SWINBURNE UNIVERSITY OF TECHNOLOGY

Agent-Oriented Decomposition (2)

- Objective-achieving decompositions means that components
  - Have own thread of control (i.e., components should be *active*)
  - Encapsulate the information and problem solving ability needed to meet these objectives.
- Since the components have only partial information about their environment
  - They need *autonomy* over their choice of action at runtime

Agent-Oriented Decomposition (3)

- Consequence of system's inherent complexity
  - interactions will occur at unpredictable times, for unpredictable reasons, between unpredictable components.
- Advantages of deferring to run-time decisions about component interactions
  - Problems associated with the coupling of components are significantly reduced by flexible and declarative modelling of component interactions.
  - Problem of managing relationships between components is significantly reduced by continuously inter-agent interactions.

Agents versus Objects

- Similarities
  - A number of similarities, e.g. both emphasise the importance of interactions between entities.
- Differences
  - Objects are generally passive in nature
  - Objects encapsulate state and behaviour, but do not encapsulate behaviour activation (action choice). Thus, any object can invoke any publicly accessible method on any other object. This modus operandi places all the onus for invoking behaviour on the client.
  - Individual objects represent too fine-grained behaviour and method invocation is a too primitive mechanism for describing the types of interactions in complex systems (design patterns and application frameworks to solve the problem).
  - Object-oriented approaches provide minimal support for structuring collectives (basically relationships are defined by inheritance class hierarchies). Complex systems involve a variety of organisational relationships (of which “part-of” and “is-a” are but two of the simpler kinds).

Agents versus Component-ware

- As components are descended from object-oriented technology, they inherit all the properties of objects
- Similarity
  - Single unit of deployment. Thus, like components, agents are typically self-contained computational entities, that do not need to be deployed along with other components in order to realise the services they provide.
- Differences
  - Components are not autonomous in the way that we understand agents to be
  - Components do not have corresponding notion of reactive, proactive, or social behaviour

In Summary

- Agents are
  - a reasonable approach to decompose a problem into multiple autonomous entities that can act in flexible ways to achieve their set of objectives
- True is
  - That agent-oriented mechanisms can be implemented using any other programming paradigm such as object-orientation or component-ware.
- However, this misses the point
  - Agent-oriented concepts introduce a different mindset for modelling complex systems and introduces extended techniques.

Agent-Oriented Software Lifecycle

- Specification
  - Specify agents' attributes such as beliefs, desires, and intentions
    - Goals - agents will try to achieve,
    - Actions - agents perform and the effects of these actions,
    - Beliefs - the information agents have about their environment, which may be incomplete or incorrect,
    - Interaction - how agents interact with each other and their environment over time.
    - E.g. Gaia, Prometheus
- Implementation
  - translate or compile the specification into a concrete computational form
  - E.g. Multiagent Systems Engineering Methodology (MaSE)
- Verification
  - Axiomatic Approaches
  - Semantic Approaches: Model Checking

SWINBURNE UNIVERSITY OF TECHNOLOGY

Situation Today

- Software Engineering approaches
  - Focus on design and architectural issues
  - Define frameworks and Protocols (e.g. FIPA)
- AI approaches are mostly impracticable for software development
- Exception JACK agents
  - (http://www.agent-software.com/shared/products/index.html)
- Lack of intelligence in today's multi agent software systems

References

- N. Jennings, M. Wooldridge, Agent-Oriented Software Engineering.
- Amund Tveit, A survey of Agent-Oriented Software Engineering.
- Katia Sycara, Multiagent Systems.

SWINBURNE UNIVERSITY OF TECHNOLOGY