

Agent-Based Transformation Architecture for Developing Intelligent Systems

Somkiat Kitjongthawonkul*

*School of Business, Australian Catholic University, 250 Victoria Parade, East Melbourne, Vic 3002, Australia

somkiat.kitjongthawonkul@acu.edu.au

Abstract— Systems development can be seen as a process of transforming users' tasks and perceptions into computer-based software artefacts. In this paper, we propose the transformation agent architecture for developing intelligent systems. We view the information system transformation ontology as comprising of content ontology and mechanism ontology. We reinforce the information systems transformation ontology concept by associating its components (content ontology and the mechanism ontology) with software models (object model, agent model, and intelligent technology model) and problem solving adapter models. The outcome of this integrated reinforcement is an agent-based transformation architecture which can be used to develop intelligent systems. The association of objects, agents, and intelligent technologies with problem solving adapters is based on their intuitive tasks and structure modelling strengths respectively, rather than their technological motivations.

Keywords— Information systems transformation ontology, Transformation agent architecture, Problem solving agent, Software artefact, Intelligent technology model, Agent model, Object model

I. INTRODUCTION

Systems development can be viewed as a process of modelling and transforming user's tasks and perceptions into some form of script that is executable in a computing environment [1]. As the objective of systems development is to provide a solution to the problem, it is essential that the systems are to be modelled with respect to the type of problem which it is built for. Therefore, two related questions need to be addressed: which problem the system should solve, and how the system is going to solve the problem.

A number of modelling techniques have been employed for such interest, see [2, 3], for example. Among some of the best known include data-flow diagrams, state transition diagrams, entity-relationship diagrams. Although these modelling techniques are excellent and widely used for modelling and designing software systems, their successful implementation is largely dependent on the experience of the system modeller who justifies the modelling techniques to be used at different levels of systems development process. Furthermore, their appropriateness for modelling modern intelligent systems that are based on the synergy between humans and computers are doubted. The principles of these modelling techniques are largely influenced by the computer world, which address the problem the system should solve from the computational end,

rather than being based on involvement of people and their tasks. In fact, in organizations, people solve problems in a task context rather than in a technology context. To this end, it is necessary to understand the way humans perform or accomplish their tasks; the things they do, the things they act on and the things they need to know [4]. The motivation of this interest comes from a number of perspectives, including a study done by [5] of major accidents in power plants, aircraft, and other complex systems. The study found that 60 to 80 percents of the accidents were blamed on operator error, mistaken or incorrect action by people who operate equipment or systems. Given the nature of human limitations, much of the blame might have gone to poor system modelling and a design that created a high likelihood of error.

We present in this paper Transformation Agent Architecture (TAA) for developing multi-media data base systems, information systems and intelligent systems. From a systems development point of view, we recognize TAA as an instance of the transformation ontology, which is one of the three ontology components of the overall information systems ontology, as shown in Figure 1. We outline the task-based problem solving adapter, which is used as a base to unify features derived from the intelligent technology model, agent model, and object model. The underlying philosophy of problem solving adapters is based on the distributed cognition approach as outlined by [6-9]. The software artefacts such as objects and agents have been integrated to the problem solving adapters as a means of realizing the characteristics of the problem solving adapters at their computational level. The selection of the software artefact is based on their intuitive modelling strengths rather than their technological motivations.

This paper is organised as follows. Section II outlines the information systems transformation ontology as well as providing a modelling view of unifying software artefacts and problem solving adapters. Section III provides information on the transformation agent architecture. Section IV briefly describes an example of applying the features of TAA. Section V, which is the last section, concludes with a summary of the paper.

II. INFORMATION SYSTEMS TRANSFORMATION ONTOLOGY

Information systems ontology provides explicit knowledge level specification of conceptual frameworks for modelling

domain knowledge which may be affected by the particular domain and task it is intended for [10]. Zuniga, in [11], defines information systems ontology as being an axiomatic theory made explicit by means of a specific formal language. Being specific to a particular domain, information systems ontology needs to account for the intended meaning of a formal vocabulary or protocol that are employed by the agents of the domain under investigation [11]. Keeping this view, we propose that information systems ontology should encompass ontological components such as activity analysis ontology, transformation ontology, and media interpretation and presentation ontology, as shown in Figure 1.

This section briefly outlines some of the features of the transformation ontology component, which involves content ontology and mechanism ontology.

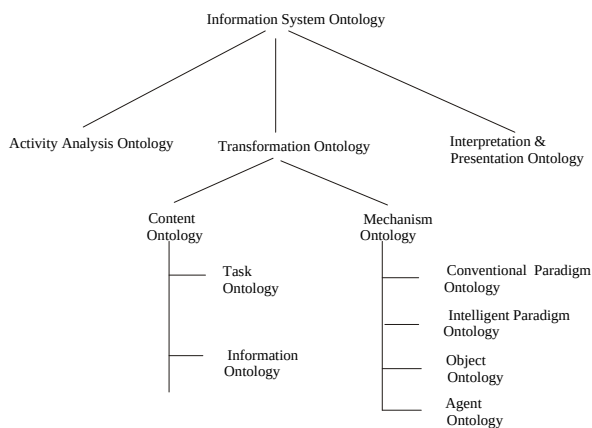


Figure 1. Information systems component

A. Content Ontology

The content ontology defines various tasks, data and information required by agents (which could be humans and/or computers) as well as information processing phases to perform their work activities. In this paper, we describe the overall content ontology using the definition of task-based problem solving adapters (PSA), as shown in Table 1. The use of PSA can be thought as analogous to a device that facilitates mapping/transforming some of the real world tasks and associated data to a computer software artefact. The PSA has taken into account a number of perspectives such as complex real world systems, neurobiological control, cognitive science, man-made physical systems, human-centeredness, learning, knowledge representation, task complexity, conscious and automated behaviour, and others. The PSA has been based on experience of developing software systems for complex real world problems like image processing, sales management, diagnosis and others. More detail of PSA can be found in [12, 13]. The adapters are described in terms of the information processing phases (namely, preprocessing phase, decomposition phase, control phase, decision phase, and postprocessing phase), tasks to be accomplished in each phase, intelligent methods used for accomplishing the tasks in different phases and hybrid

configuration of the intelligent methods. Some of the features of the PSAs are summarized in Table 1.

The intelligent methods are selected based on top down and/or bottom up knowledge engineering strategies. The intelligent mechanisms reflect integration of artificial and computational intelligent levels of problem solving. This involves symbolic, fuzzy, neural network, and genetic algorithm methods. The architecture incorporates fusion, transformation and combination methods as a mean to maximize both the quality of solution and range of tasks to be handled. In situations where the nature of the problem is rather structured and/or semi-structured orientation, the intelligent methods and hybrid arrangement may not necessary. In contrast, however, these intelligent mechanisms are required if the problems are unstructured.

TABLE 1. SUMMARY OF TASK-BASED PROBLEM SOLVING ADAPTER

PHASE	(SOME) TASKS	INTELLIGENT METHODS (optional)	HYBRID CONFIGURATION
Preprocessing phase	Eliminates noise which is peculiar to problem under study, input conditioning	Symbolic Methods, Fuzzy Methods	Transformation
Decomposition phase	Analyzes and decomposes the problem domain into set of abstract concepts/classes	Symbolic Methods Neural Network Methods	Combination
Control phase	Operates within each abstract class and performs input validation and problem formulation, determination of decision level classes, and resolving conflicting outcomes of decision level classes	Symbolic/Fuzzy Methods, Neural Network Methods, Fuzzy-Neural Network Methods, Genetic Algorithm Methods	Fusion, Transformation
Decision phase	Determines/predicts specific classifications with each decision-level class and provides desired outcomes to problem solver	Neural Network Methods, Numerical Methods Mathematical Algorithms, Fuzzy-Neural Network Methods, Genetic Algorithm Methods	Fusion, Transformation
Postprocessing phase	Validates decision, explains and presents decisions made in the decision phase	Symbolic/Fuzzy Methods, Neural Network Methods, Genetic Algorithm Methods	Fusion, Transformation

B. Mechanism Ontology

The mechanism ontology provides a view on using information processing mechanism that can operate on the tasks, data and information represented in the content ontology. Such information processing mechanism can be done by using the four software paradigm ontologies (e.g. conventional paradigm [2, 14, 15], intelligent paradigm [16-19], object paradigm [20-22], and agent paradigm [23-26]) individually or in unison. Indeed, it could be viewed that the mechanism ontology complements the content ontology to facilitate constructing a software artefact at the computational level architecture of the transformation ontology.

III. TRANSFORMATION AGENT ARCHITECTURE

In this section, we solidify the transformation ontology by merging software models with the problem solving adapter model as a frame. To this end, the characteristics of the three software models are associated with the problems solving adapter model. These three software models consist of the object model, agent model, and intelligent technology model. Figure 2 shows the association of the models. Various features of object-orientation including inheritance, composability and other non-hierarchical relationships between various concepts facilitate structuring representations of the problem domain class, agent class and problem solving class. This allows development of generic data structures and methods, a multiple inheritance structure and formation of heterogeneous objects.

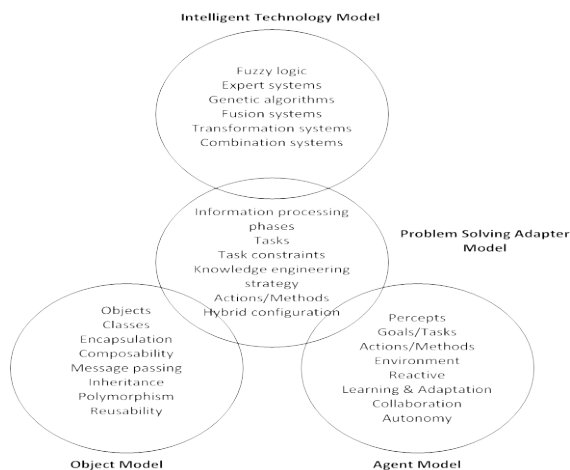


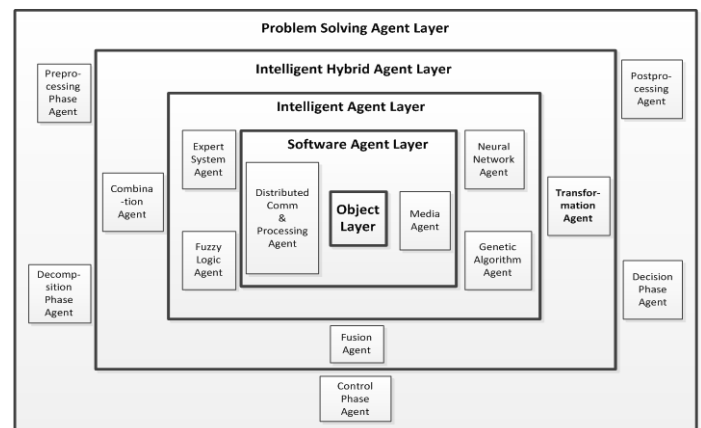
Figure 2. Association of object, agent, intelligent technology, and problem solving adapter models

The notion of agents used here is derived from the agent and agent architectures of [19]. As agent is based upon a task/goal abstraction in which an agent engages in the problem domain. Thus, it is used for mapping the task context of PSAs to represent the structure of the problem solving agents in terms of PAGE description [19]. This is in contrast to object-orientation which is an entity that represents some data items in the system together with the operations that can be performed on the data item during system activity. An agent facilitates task-based communication and defines the nature of communication between different agents which can be intricate and complex (e.g. negotiation) depending upon the application domain. The features of agent-orientation help us to capture goal/task-based orientation which is dynamic by nature. In addition, the properties of the agent such as learning and adaptation are especially important for describing the human behaviour aspect which are tasks or goals driven in an appropriate way. The intelligent technology model, which consists of expert systems, fuzzy logic, neural networks, genetic algorithms, along with their intelligent hybrid configurations, helps to capture various intelligent technologies used for modelling different data and tasks.

The PSA is used as a frame of integration of object and agent models to enable effective realization of task context in which humans solve problems. The role played of the PSA is then providing agents the intelligent strategies and methods for achieving or satisfying agents' tasks/goals under given constraints such as fast response time, reliability and temporal reasoning. This is done by executing a sequence of actions through the five information processing phases, and tasks associated with each phase and intelligent methods used to accomplish the tasks in each different phase.

The association of object model, agent model, intelligent technology model, with task-based problem solving adapter model forms a building block of Transformation Agent Architecture (TAA) shown in Figure 3.

Figure 3. Building block of transformation agent architecture



As shown in Figure 3, TAA is built on five layers. The object layer, software agent, intelligent agent, intelligent hybrid agent, and problem solving agent layers respectively. The role of the object layer, which is the inner layer, is to primarily capture problem domain objects and define their structural relationships. The software agent layer consists of the generic distributed process agent, communication software agent, and media agent. The intelligent agent layer consists of the generic expert system agent, generic supervised/unsupervised neural network agent, generic fuzzy logic agent, and generic genetic algorithm agent. The intelligent hybrid agent layer makes use of intelligent features and methods provided by the intelligent agent layer (e.g. those of the expert system agent, fuzzy logic agent, and genetic algorithm agent). The intelligent features and methods can be utilised based on the fusion, transformation, and combination of configurations. Finally, the outer layer called the problem solving agent layer consists of five generic agents namely preprocessing, decomposition, control, decision, and postprocessing respectively. These agents communicate with each other by means of sending and receiving messages using appropriate communication constructs such as request, inform, order, etc. All the agents in the architecture appear in Figure 3 are defined based on the generic agent definition shown in Table 2.

TABLE 2. GENERIC AGENT DEFINITION TEMPLATE

Name:
Parent agent:
Goals:
Tasks:
Task constraints:
Precondition:
Postcondition:
Communication with:
Communication constructs:
Represented features:
Internal tools:
Internal state:
Action:

IV. APPLICATION

This section gives an illustrative example of applying the TAA concepts to develop an Order Monitoring System (OMS) for sales and marketing function in an electrical manufacturing company. The company has a network of branches and manufacturing divisions to market and supply electrical equipment requirements for various types of industrial customers, e.g. original equipment manufacturers, contractors, consultants, and small private industrial customers. The main products include a wide range of Luminaries, Motors, Alternators, Transformers, Switchgear, LT Panels, MCCBs (Moulded Case Circuit Breakers) Capacitors, etc. Order processing is a backbone function of the company. Its execution spans across various organizational functions including sales and marketing, financial and accounting, engineering, and production. The reason for order monitoring development is to solve the coordination problems between marketing engineers and manufacturing divisions, and the consequential delay in shipment leading to loss of future orders.

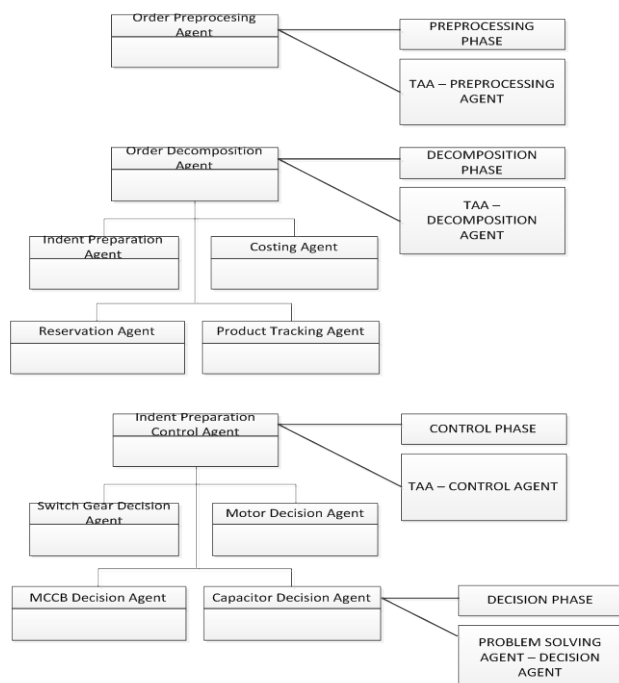


Figure 4. OMS Tranformation agents

OMS helps marketing engineers in the electrical manufacturing company to keep track of customer order processing stages like indent preparation, costing, reservation, production status, etc. An overall of transformation agents employed in the OMS is shown in Figure 4. The task context performed by the stakeholders of the OMS (e.g. marketing manager and marketing engineers) to solve their problems is captured using the generic transformation agent definition template, which is shown in Table 2.

TABLE 3. DEFINITION OF INDENT PREPARATION AGENT OF THE OMS

Name:	Indent Preparation Agent
Parent agent:	TAA Control Agent
Goals:	Monitor the preparation of Capacitor, Motor, Transformer indents; Monitor changes in the indents; Coordinate the information with indents
Tasks:	Indent preparation data validation; Parsing incoming message; Error report; Conflict resolution between decision agents
Task constraints:	Efficient, Simple query; Domain dependent
Precondition:	Indent preparation data; Customer-related data; Product-related data
Postcondition:	Order accepted; Order rejected
Communication with:	Order Processing Decomposition Agent; Order Writing Decision Agent; Order Assembly Agent; Invoice Processing Agent
Communication constructs:	Request: Formatted Order writing data Command: Order Decision Reply: Process state Send: Report
Represented features:	Text and numeric data will be transformed to the valid format, and organised as single data
Internal tools:	Planning and sequencing of actions related to order writing
Internal state:	Order; Product; Customer
Actions:	Order data validation rules Check order type Check indents Distributed communication rules Conflict resolution rules Send formatted data Reply Parsing Action sequence rules

All the transformation agents in Figure 4, which are the Indent preparation agent, Costing agent, Reservation agent, and Product tracking agent. All are well equipped with agent-orientation features such as percept, goals/tasks, reactive, collaboration, etc. These features facilitate collaborative problem solving among these agents to achieve their goals/tasks. For example, in Table 3, the goals of the Indent preparation agent in the context of OMS are: a) to monitor the preparation of Capacitor, Motor, and Transformer indents, b) to monitor changes in the indents, and c) to coordinate the information with indents. To achieve these goals, the Indent preparation agent needs to engage a number of tasks including establishing communication constructs with other agents and objects, for example the Capacitors indent agent, Motor indent

agent, Transformer indent agent, and Order object, in forms of request, inform, and command. Messages are sent and/or received via an interface channel of the Indent preparation agent. The messages perceived by the Indent preparation agent are mapped internally into a series of actions, such as identifying products/articles, checking order types, and checking indents. It is noteworthy that the communication acts establish virtual communication links between tasks of the indent preparation agent and the services offered, by linking to this agent in OMS object structure. These objects are Customer, Order, Product technical document, Indent form and Product catalogue. Due to space limitations, the exhaustive features of the transformation agents used in OMS cannot be adequately covered. However, more detail about modelling and design of OMS can be found in [12].

V. CONCLUSION

The Transformation Agent Architecture (TAA) described in this paper consists of five layers (one object layer plus four agent layers). The role of the object layer is primarily to capture the problem domain objects and define their relationships. The four agent layers provide a range of capabilities related to different aspects of problem solving. The problem solving agent layer provides capabilities to model user's tasks in a way which is separated from the low level details of various techniques and methods (e.g. neural networks) employed to execute various actions. The intelligent hybrid agent layer and intelligent agent layer provide intelligent tools of varying complexity to satisfy various tasks constraints in a problem domain. The software agent layer facilitates communication between problem solving agents as well as providing them with media related services such as display and layout. The agents in the four agent layers can be used in isolation or in unison. Table 2 provides a generic agent definition temple for defining these agents.

To illustrate how the TAA approach can be realised in a real problem domain, a brief discussion about applying the TAA approach to develop a system for monitoring sales stages of a large electrical manufacturing company was provided in this paper.

Work on the integrated system development framework is in progress, as it is not yet entirely complete. This will be included in future papers, as well as providing a new direction for further research.

REFERENCES

- [1] Y. Wand, D. Monarchi, E., J. Parsons, and C. Woo, C., "Theoretical foundations for conceptual modelling in information systems development," *Decision Support Systems*, vol. 15, pp. 285-304, 1995.
- [2] D. Avison and G. Fitzgerald, *Information Systems Development: methodologies, techniques and tools*, 3rd. ed.: McGraw-Hill, 2003.
- [3] J. Fortune and G. Peters, *Information Systems: achieving success by avoiding failure*: John Wiley & Sons, Ltd, 2005.
- [4] A. Dix, J. Finlay, G. Abowd, and R. Beale, *Human-Computer Interaction*: Prentice Hall International (UK) Limited, 2003.
- [5] C. Perrow, *Normal accidents: living with high-risk technologies*. NY: Basic Books, 1984.
- [6] D. A. Norman, *Things That Make Us Smart*. Reading, MA: Addison-Wesley, 1993.
- [7] J. Zhang, "Nature of External Representations in Problem Solving," *Cognitive Science*, vol. 21, pp. 179-217, 1997.
- [8] M. Wagman, *Cognitive Science and Concepts of Mind: Toward a general theory of human and artificial intelligence*. New York: Praeger Publishers, One Madison Avenue, 1991.
- [9] B. Blum, I., *Beyond Programming: to a new era of design*. New York: Oxford University Press, Inc., 1996.
- [10] G. van Heijst, A. T. Schreiber, and B. J. Wielinga, "Using Explicit Ontologies in KBS Development," *International Journal of Human and Computer Studies*, 1996.
- [11] G. L. Zuniga, "Ontology: Its Transformation From Philosophy to Information Systems," in *FOIS '01 2nd International Conference on Formal Ontology in Information Systems*, Ogunquit, ME, USA, 2001, pp. 187-197.
- [12] J. Ko and S. Kitjonthawonkul, "Task-Based Ontology of Problem Solving Adapters for Developing Intelligent Systems," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 4, pp. 353-360, December 2004 2004.
- [13] S. Kitjonthawonkul and R. Khosla, "Task-Based Ontology for Information Systems," in *Proceedings of the 2000 Americas Conference on Information Systems (AMCIS 2000)*, Long Beach, California, USA, 2000, pp. 317-323.
- [14] YOURDON Inc., *Yourdon Systems Method: model-driven systems development*. Englewood Cliffs, New Jersey: PTR Prentice Hall, A Simon & Schuster Company, 1993.
- [15] P. P. S. Chen, "The Entity-Relationship Approach," in *Information Technology in Action: trends and perspectives*, R. Y. Wang, Ed., ed New Jersey: PTR Prentice Hall, 1993, pp. 13-36.
- [16] E. Davlo and P. Naim, *Neural Networks*. UK: Macmillan Computer Science Series, 1991.
- [17] J. J. Grefenstette, "Genetic Algorithms and their Applications," in *Encyclopaedia of Computer Science and Technology*. vol. 21, A. Kent and J. G. William, Eds., ed Washington DC: Naval Research Laboratory, 1990, pp. 139-152.
- [18] K. Hamada, T. Bab, and M. Yufu, "Hybridizing a Genetic Algorithm with Rule-Based Reasoning for Production Planning," *IEEE Expert*, pp. 60-67, 1995.
- [19] S. Russell and P. Norvig, *Artificial Intelligence: A modern approach*, 3rd. ed. New Jersey: Prentice-Hall, Inc., 2009.
- [20] S. Clarke and R. J. Walker, "Composition patterns: an approach to designing reusable aspects," in *Proceedings of the 23th International conference on Software engineering*, 2001, pp. 5-14.
- [21] F. V. Assche, B. Moulin, and C. Rolland, *Object oriented approach in information systems*. Amsterdam, The Netherlands: Elsevier Science Publishers B.V., 1991.
- [22] M. Blaha and J. Rumbaugh, *Object Oriented Modeling and Design*: Pearson Prentice Hall, 2005.
- [23] M. Wooldridge and N. R. Jennings, "Agent Theories, Architectures, and Languages: A Survey," in *ECAI-94 Workshop on Agent Theories, Architectures, and Languages*, Amsterdam, The Netherlands, 1994, pp. 1-39.
- [24] M. Wooldridge, "An Introduction to Multiagent Systems," ed: John Wiley & Sons, Ltd., 2002.
- [25] R. Pfeifer and P. F. M. J. Verschure, "Distributed Adaptive Control: A Paradigm for Designing Autonomous Agents," in *Proceedings of First European Artificial Life Conference*, Cambridge, MA, 1992, pp. 21-30.
- [26] H. S. Nwana and M. Wooldridge, "Software Agent Technologies," in *Software Agents and Soft Computing: Towards enhancing machine intelligence; concepts and applications*, H. S. Nwana and N. Azarmi, Eds., ed Berlin: Springer- Verlag, 1997, pp. 59-78.