Design of Agent-Oriented Pattern Templates

Ayodele Oluyomi Shanika Karunasekera Leon Sterling
Department of Computer Science and Software Engineering
The University of Melbourne
111 Barry Street, Carlton, Victoria 3053, Australia
{aoo, shanika, leon}@cs.mu.oz.au

Abstract

Software patterns have potential to enhance the comprehension, application and communication of agent oriented software concepts. To realize this potential, agent oriented patterns should be appropriately described with the aid of good pattern templates. However, many of the existing descriptions of agent patterns lack certain features that are critical to the description of agent patterns. In this paper, we present a design for agent oriented pattern templates. We present two templates built according to our design that we believe improve on the description of agent patterns. The InteRRaP agent architectural pattern and Contract Net agent interaction pattern are presented using these templates.

1. Introduction

Software patterns have been used, especially in the object-oriented programming community, to communicate important experience within software development. In the agent based development context, patterns are useful for connecting conceptual modeling with implementation of agent systems [10] and communication of agent concepts [7]. We believe that to facilitate the wide uptake of agent technology by industry, there needs to be the ability to share experience of what is and is not successful for agent development.

A pattern needs to be appropriately described for it to be an effective tool of experience sharing. The pattern template is the guide for describing patterns. The description of a pattern should contain only necessary but adequate information of the experience to be shared. An inappropriate or inadequate pattern template can seriously affect the usefulness of patterns.

From the examination of different agent pattern templates (see section 2), we observe most agent patterns do not reflect the concepts of agency in the pattern templates. Also, features such as problem, context, forces considered crucial to the universal description and communication of patterns, are mostly absent.

In this paper, we present a design for agent pattern templates that we believe improve on the description of agent patterns. In our previous work [15], we have argued the need to for having different templates for the different levels of abstraction in the agent oriented design process. We therefore use our design to build two templates, one for agent architectural patterns and the other for agent interaction patterns.

The paper is organized as follows. Section 2 describes related work and problems with existing pattern templates. In section 3, we present our design for agent pattern templates and two templates using this design. Section 4 presents the use of our proposed templates to describe one agent architectural pattern and one agent interaction pattern, InteRRaP and Contract Net respectively. Section 5 compares our templates with those of two other approaches. Section 6 concludes and presents our directions for further work.

2. Related Work

The pattern template (also known as pattern form or pattern description) is the vehicle for conveying the experiences that patterns are meant to share. A good pattern description should include a specific recurring *problem* in a *context* that defines a set of *forces* which are resolved by a general *solution* to create a *resulting context*.

Different pattern templates have been used in describing agent oriented patterns. Chacon et al [3] present two patterns without using any template. Tahara et al [5] describe a set of patterns with a three element template. Sauvage [6] uses a pattern template with eight elements to describe the patterns presented. Weiss [7] presents a pattern language describing the

Proceedings of the 2006 Australian Software Engineering Conference (ASWEC'06)
1530-0803/06 $20.00 © 2006 IEEE

individual patterns with a pattern template of five elements similar to the basic Alexandrian form. Aridor and Lange [8] use an abridged (six elements) GoF form [9] in describing their agent mobility patterns. Schelfthout et al [10] present their agent implementation patterns using an adapted version (seven elements) of the GoF form. Sandra et al [11] described patterns for multiagent coordination using the GoF template.

These researchers use different pattern templates. However, they do not give a rationale for the chosen template or for adapting existing templates. The lack of rationale for defining these templates makes it difficult to compare the patterns; combine the patterns into pattern languages; or integrate the patterns into existing design.

While agent oriented patterns templates should describe the basic features of a good pattern (i.e. problem, context, forces, solution, and resulting context) [2], they require additional features in order to capture the notions of agency effectively. Heinze presents some agent patterns in [12] and states that these agent patterns require a more sophisticated description template. Also, Lind [13] asserts that agent oriented patterns require more complex templates.

Lind argues that a single general template is inadequate for agent patterns [13] and proposes a two part pattern description template having a general part and a view-specific part. However, the view specific part mixes elements related to problem with solution and forces. Therefore, there is no coherent flow in the description of the pattern presented with such a template. Malyankar [14] proposes a pattern template for social structures in agent systems by adding elements and sub categories to the GoF template, however, the modifications do not reflect the notions of agency.

A review of these templates in describing agent oriented patterns reveals the following inadequacies.

- Most existing templates lack agent concepts. Some of the templates that do have agent concepts do not structure them in a way that maintains the features of a good pattern.
- Lack of clear rationale for the introduction of template elements or shortening existing templates make it difficult to follow the structure and message of the pattern.
- The assumption that one template is good for all categories of patterns i.e. they are not tailored to particular levels of abstraction
- Ambiguity in the meaning of template elements. For example Intent and Motivation in GoF template, see section 5 for more discussion of this.

We argue that an agent pattern template should be designed with template elements that capture the notions of agency and according to the levels of agent system development. The design should also be in line with the main features of a good pattern to facilitate communication, adaptation and integration. We present a design for agent oriented pattern templates and two templates built using this design, in grounding these arguments.

3. Approach to Pattern Template Design

In this section, we present our design for agent oriented pattern templates. We also present templates for agent architectural and agent interaction patterns based on this design.

Our design for agent pattern templates is based on three premises. These are:

1. Template structure should be in line with the main features of a good pattern. That is, the elements of the template should be structured to clearly define the problem, context, forces, solution, and resulting context of the pattern [2]. Structuring patterns in this way will facilitate universal communication of the concepts described in the pattern.
2. Pattern templates should be designed to ensure that the notions of agency are captured in pattern descriptions. Template elements should be designed to ensure that the agent concepts are appropriately described in the pattern.
3. There should be different templates for the different categories [15] of agent patterns. A single template is not adequate for all types of agent patterns. This is because the notions of agency are represented differently at the different levels of abstraction of agent based development. Hence, the template elements should reflect these differences.

Given the premises above, we design an agent pattern template to consist of major template elements and sub-elements. The sub-elements are combined with the major elements to generate a template structure for a particular category of agent patterns. The major template elements are the common features of a good pattern i.e. problem, context, forces, etc. The sub-elements are introduced to capture the different concepts of agency necessary for describing an agent pattern.

All categories of agent patterns according to our design have the same set of major template elements. However, the sub-elements vary from category to category according to the relevant concepts of agency for each category. See Table 1. This structure ensures clarity in template design; it facilitates communication with non agent practitioners and enhances adaptation and integration of agent patterns.

Sub-elements considered appropriate for a particular category of patterns are defined by carrying out two tasks. First is a study of the different levels of agent oriented design. This study is used to define the design issues and features of software agents peculiar to each level [15]. Second is the examination of some agent oriented design patterns (section 2) in order to find out what information is crucial to the description of agent patterns.

The emphasis of our design is on the Forces and Solution elements. A good pattern is one that describes how a general solution is able to resolve a system of forces. Hence, to improve the pattern template, our emphasis is to introduce agent concepts as sub elements to Forces and Solution. Laying out the forces explicitly helps to describe the unresolved forces in the resulting context.

3.1 Template for Agent Architectural Patterns

We present a template for describing patterns at an agent architectural level. We include definitions of the sub-elements that we consider essential in describing agent architectural patterns.

Name: Describes a name for referring to the pattern and other names by which the pattern is described, if any.

Classification: Describes the position of the pattern in the two way classification according to [15].

Problem: Defines the recurring problem that the general solution is defined to solve.

Context: Describes a particular setting for the problem which provides the basis for defining the relevant forces.

Forces: Describes the constraints that are relevant to a particular problem based on the context of the problem. The following agent concepts are introduced as sub elements to the Forces element.

Goal defines commitment to goals; hierarchy of goals; stability of goals, etc.

Autonomy defines the degree of dependence on client for final decisions; and degree of control over own services

Social ability defines the interaction related constraints. Agent interaction may require negotiation strategies; secured messages e.g. in military applications, and so on.

Environment defines the nature of the agent's environment.

Adaptive behaviour defines the need to be able to sense changes in agent's usage trends and behaviour of the environment and modify behaviour accordingly.

Intelligence defines the constraints that relate to the amount of knowledge to be used in decision making.

Decision and action defines the level of accuracy and urgency that is required in decision making. Also, it defines constraints related to action utilities and costs.

Solution: Describes the general solution that best resolves the forces identified. We introduce the following sub elements to capture the notions of agency in the template:

Control coordinates the other components of the architecture. It may stand alone or be part of another component. Control handles issues like choice of reactive over deliberative actions (or vice versa). It also handles the autonomy of the agent.

Strategy addresses two issues. One is goal definition strategy i.e. how the agent decides the goals to commit to. The other is action execution strategy i.e. how the agent achieves the goal(s). Action execution strategy addresses the decision on reactive or deliberative behaviour or both.

Knowledge Management addresses what constitutes domain knowledge; knowledge updates; and decisions about persistence of knowledge.

Interaction Management handles interaction issues like type of interaction; strategy for interaction; message interpretation, interaction protocols and so on.

Environmental Interface addresses the perception, action and message communication capabilities and functions of the agent.

The following sub elements have been used in existing pattern templates; however, we classify them as sub elements under Solution to maintain our structured approach of template design.

Structure defines the arrangement of the component parts of the architecture. It is usually presented as a diagram.

Dynamics describes how the component parts relate in order to realize the *Strategy* of the architecture.

Known uses: Specifies existing applications of the pattern.

Resulting context: Describes the effects of applying the general solution to the initial context/forces, the advantages of the solution and unresolved forces. We introduce one sub element as follows.

Adaptation/Integration presents suggestions on how to adapt the pattern to differing projects and how to combine it with other patterns to generate agent pattern languages.

Related Patterns: Specifies patterns that lead to this or those that follow from the application of this or patterns that are alternates to this pattern.

See Table 1 for a summary of this template.

3.2 Template for Agent Interaction Patterns

We present a template for describing agent interaction patterns. We include definitions of the sub-

elements that we consider essential in describing agent interaction patterns.

Name: Describes a name for referring to the pattern and other names by which the pattern is described, if any.

Classification: Describes the position of the pattern in the two way classification according to [15].

Problem: Defines the recurring problem that the general solution is defined to solve.

Context: Describes a particular setting for the problem which provides the basis for defining the relevant forces. The following sub element is introduced to elaborate description of context for agent interaction patterns.

Goal: defines the system goal that requires the interaction e.g. optimize investments.

Forces: Describes the constraints that are relevant to a particular problem based on the context of the problem. The following agent concepts are introduced as sub elements to the Forces element.

Environment: (has a different meaning to what it represents under architecture template) defines the nature of other agents that will be involved in the interaction. E.g. malicious agents, cooperating agents, competing agents, etc. It defines the environment is static or dynamic. It also specifies information about access to the participants i.e. it states if the addresses of the participants in the interaction are known by all the agents in the MAS.

Adaptive Behaviour: specifies adaptive behaviour of the MAS and how this impacts on the interaction amongst the agents. For example, a self organizing Smart Home requires adaptive interaction.

Security: specifies the sensitivity of the interaction in question, to the safety/security of the MAS. For instance, interaction in a Smart Home in case of a burglary should be secure. It also defines the impact of time of interaction on the security and in determining the success of the MAS. For instance, safety and security in Air Traffic Control may be time critical while interaction in an every day trading transaction may not be time critical.

Language constraints: specifies the uniformity or otherwise of the participating agents' representation of the real world.

Solution: Describes the general solution that best resolves the forces identified. We introduce the following sub elements to capture the notions of agency in the template:

Interaction Strategy: specifies the type of interaction. It could be a simple type such as negotiation, collaboration or a composite type such as competitive-negotiation. It also specifies the policies guiding the conduct of the interaction. For instance, inclusiveness considerations can be used to handle the issue of malicious agents.

Structure: defines two things. One, participants i.e. the agents that will take part in the interaction including new agents introduced into the MAS to facilitate the interaction. Two, it defines the static arrangement of the agents to be involved in the interaction.

Collaboration/Dynamics: describes the actual performance of the interaction according to this pattern.

Ontology: specifies strategy for representation of the real world for the purpose of the interaction where the participants have different representations of the real world. E.g. use of a neutral representation of the real world that the participating agents must interpret

Known uses: specifies existing applications of the pattern. It is useful for developers to have further insight into the application of the pattern in real life situations.

Resulting context: Describes the effects of applying the pattern to the initial context, that is, the state of the system after the application of the pattern. It describes how the forces are resolved as well as what new forces and/or entities get introduced by applying the pattern. We introduce two sub elements for defining the resulting context.

Introduced entities: some interaction patterns introduce new agents into the system to basically facilitate the interaction. Such new agents are described here.

Adaptation/Integration: other aspects of design that this pattern is expected to impact and how this pattern can be combined with other patterns. The intent is to facilitate the finding of related patterns and building a pattern language across the entire design space.

Related Patterns: specifies the patterns that are related to this pattern, either through direct application or being alternate ways of expressing the knowledge encapsulated by the pattern.

4. Examples

This section describes two examples of agent patterns, namely InteRRaP, an agent architectural pattern and Contract Net, an agent interaction pattern. We present as many details as possible in describing these patterns, subject to space limitations.

4.1 Name: InteRRaP Architecture [1, 18, 19]

Classification: Agent Architectural level

Problem: How to build agent architectures that combine the advantages of reactive and deliberative architectures?

Table 1: Structure of Agent Pattern Templates

Major Template Elements	Sub-elements for agent architectural patterns	Sub-elements for agent interaction patterns
Name		
Classification		
Problem		
Context		▪ Goal
Forces	▪ Goal ▪ Autonomy ▪ Social Ability ▪ Environment ▪ Adaptive Behaviour ▪ Intelligence ▪ Decision and action	▪ Environment ▪ Adaptive Behaviour ▪ Security ▪ Language Constraint
Solution	▪ Control ▪ Knowledge Management ▪ Strategy ▪ Interaction Management ▪ Environmental Interface ▪ Structure ▪ Dynamics	▪ Interaction Strategy ▪ Structure ▪ Dynamics ▪ Ontology
Known Uses		
Resulting context	▪ Adaptation/ Integration	▪ Introduced Entities ▪ Adaptation/ Integration
Related Patterns		

Context: Designing agent architectures for a process environment with distributed control and joint planning (e.g. flexible production systems).

Forces:

Goal

- Goals are presented in a hierarchy. For example, a goal of maximizing profit may have sub goals of either minimizing cost or maximizing sales volume.
- A choice has to be made from the alternative sub goals.

Autonomy

- Distribution of control is required.
- Timely decision making resulting from changes in operating conditions required.

Social Ability

- Extensive collaboration with other agents required for cooperative planning
- Interactions are flexible and generalized.

Environment

- Environment is relatively closed and observable
- Environment is relatively deterministic.
- The environment is non-episodic. There is a connection between different scenarios.

Adaptive Behaviour

- The agent has to operate in a self organizing system in response to different environmental changes.

Intelligence

- High degree of flexibility in interaction and processing is required
- Optimization of time and resources is critical.

Decision and action

- The decisions and actions should be accurate.

Solution: The InteRRaP architecture is composed of a world interface, a knowledge base and a control unit.

Control: The flow of control in InteRRaP is vertical. It is a 2-pass vertically layered architecture with three layers of control which are Behaviour-based Layer (BBL), Local Planning Layer (LPL) and the Cooperative Planning Layer (CPL). CPL and LPL are the deliberative layers while the BBL is the reactive layer. Information flows from lower level layers to higher level layers while control flows from high level layers to lower level layers. InteRRaP achieves a balance between reactive and deliberative behaviour by using bottom-up activation and top-down execution.

Knowledge Management: This component of InteRRaP is made up of three layers which are the World Model, Mental Model and Social Model. These layers hold information that corresponds to the BBL, LPL and CPL respectively. Each of these models has a Belief Revision function (BR) for updating the information in the layer.

Strategy: InteRRaP uses two general functions in each of the control layers. The *situation recognition and goal activation* (SG) function is used to generate the goals to commit to. The *planning and scheduling* (PS) function is used to derive the plans for executing the goals committed to. The three control layers cooperate to achieve a mix of reactive and deliberative behaviours. The behaviours of the agent are carried out in the BBL by the activation of reactor and procedure Patterns of Behaviour (PoBs). The LPL performs domain dependent planning. The CPL cooperates with other agents to generate joint plans.

Interaction Management: Interaction management is centered on joint planning.

Environmental Interface: The World Interface component is used for all interaction with the agent's environment. Only the BBL and world model layers have direct connection with the world interface component. All necessary client interface and communication with other agents are implemented in this component.

Structure: The InteRRaP components are structurally arranged as presented in figure 1.

Dynamics: Information from the environment is received into the world model layer of the knowledge base through the world interface component. The world model is updated with the received information using the belief revision function. The information is then passed on to the mental model and the social model layers. The updated world model is also passed to the

SG function of the BBL which uses it to generate a new set of goals (if necessary). The new set of goals is sent to the PS function of the BBL and appropriate reactor PoBs are activated. If the new set of goals cannot be handled by the PS function of the BBL, bottom-up activation is used to transfer control to the LPL (and then to the CPL). The LPL will then use top down execution to pass appropriate plans (probably from the CPL) to the BBL for execution through appropriate procedure PoBs.

Known Uses: These include FORKS [18], a system for simulation of an automated loading dock. FLIP [17] (FLexible Inter Processing).

Resulting Context

Planning: the planning capability is a major feature of the InteRRaP architecture.

Environment: it may not function optimally in an open environment with unpredictable and uncontrolled exposure to changes in the environment.

Competition: it may not function optimally in a competitive environment with possible malicious agents.

Adaptation/Integration

- The separation of the knowledge base from the control unit facilitates ease of adaptation.
- A control layer with its corresponding knowledge base could be separated and adapted. The BBL and world model are separated and adapted in the application of InteRRaP to the FLIP system [17].

Related Patterns: BDI architecture [4]

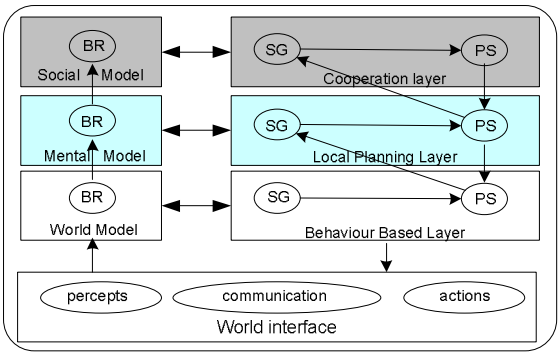

Figure 1: The InteRRaP agent architecture

4.2 Name: Contract Net [20, 16]

Classification: Agent Design Level

Problem: How should an agent execute a task that requires other agents to perform?

Context: A multi agent system where an agent requires the services of one or more other agents to carry out a task.

Goal: To optimize a function (e.g. price) that characterizes an agent's task.

Forces:

Environment: The agents are non malicious agents; there is a degree of competition amongst the service provider agents; the client agent has knowledge of addresses of the provider agents.

Adaptive Behaviour: There may be changes in individual agents' internal business rules over time.

Security: The interaction in the MAS does not significantly impact on the security of the system. Time is usually not a critical consideration.

Language Constraint: There are no language constraints. The agents in the MAS have the same representation of the real world as it relates to the task to be carried out

Solution: The Contract Net pattern approaches this problem by assigning the role of a manager to the agent that owns the task to be carried out. The manager agent sends a request to the potential contractors (other agents that can carry out the task). The proposal from the contractors are assessed by the manager and the task is awarded to one (or more) of them that best optimizes the characteristic function of the task. The contractor agent(s) then send the result (or no result) of the task execution to the manager agent and the interaction is ended.

Interaction Strategy: The Participants in Contract Net include one Manager Agent (task owner) and a known number of Contractor agents (probable task executors). The manager agent has direct interaction with each of the contractor agents. The contractor agents do not interact with one another.

Structure: Contract Net is a negotiation type interaction. The time for the presentation of proposals is deadline controlled. The participants must respond to every message sent. Contractor agents must commit to executing the task awarded to them.

Dynamics: The manager agent sends out a call for proposal to the probable contractor agents. Each contractor agent then sends a proposal for the task execution to the manager agent within a stipulated time frame. The manager agent responds to all the proposals by either accepting or rejecting them. Each contractor agent whose proposal is accepted sends the result of the task execution to the manager client and the interaction ends. See figure 2 for the dynamics of Contract Net.

Ontology: The agents involved are assumed to have the same representation of the real world as it relates to the task they want to perform. Therefore, Contract Net does not have ontological specifications.

Known Uses: Intelligent Manufacturing [21, 22]

Resulting Context: This solution does not cater for capturing the adaptive nature of the MAS as it affects interaction; there is no provision for handling language constraints if it exists among the participating agents.

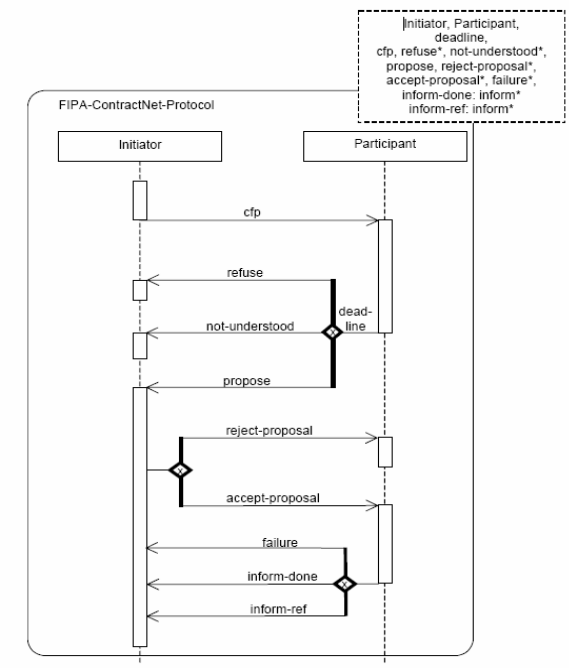

Figure 2. The FIPA Contract Net Protocol

Introduced Entities: Contract Net does not introduce agents into the MAS in order to solve the interaction needs

Adaptation/Integration: The relative simplicity of Contract Net allows for easy adaptation. Examples of adaptations of Contract Net include the Iterated-Contract Net [16]. Contract Net could be integrated with other patterns e.g. Embassy pattern [11] in a case where manager and contractor agents have different representations of the world.

Related Patterns: Broker [11], Request [16]

5. Discussion

We discuss the design for agent pattern templates presented in this paper by comparing them with two other pattern template designs.

The comparison is based on the qualities of a good agent pattern template adapted from the qualities of a good pattern [2]. These qualities are as follows. An agent pattern template should: indicate the level of abstraction it applies to; use appropriate constructs (roles, hierarchies, behaviour, proactive, etc) at the right level of abstraction; reflect the notions of agency (autonomy, proactiveness,); address the appropriate software engineering issues (implementation, organization structure,) at the relevant levels of development; enhance connection with relevant lower levels of agent oriented development; use non ambiguous elements and with no overlapping meanings.

We compare the GoF template and the View oriented template [13] with our template. The GoF template is chosen because a number of the agent pattern writers use the template and different variations of it (see section 2). The View oriented template is chosen because it has representations of agent concepts. Table 2 is a listing of the elements of the three templates compared.

The GoF assumes the same template for all categories of patterns. The View oriented approach has different templates for the two categories of patterns examined. However, it does not present a precise structure for organizing these elements. Our approach presents a structure for organizing template elements by defining a relationship between the features of a good pattern and agent category specific sub-elements. Detailed comparison follows.

GoF: The Intent and Motivation elements of the template are ambiguous and their meanings overlap. While some agent pattern writers use Intent to describe the problem [8], some use it to describe the solution [11]. [10] adapts the template by combining Intent and Motivation.

The template does not have direct mapping to problem, context, forces, solution and resulting context. This absence of structure impairs communication of patterns described by different writers; and makes adaptation of the patterns difficult.

The template assumes a single template is general enough for all categories of patterns. This is not feasible for agent oriented patterns [13, 14]

The template does not reflect the concepts of agency.

View oriented: This template is not structured in line with the basic features of a good pattern (problem, context, solution,). There is no Context element; the Dependencies element introduces ambiguity since it could either describe Forces or Solution.

Implementation is included in the general part of this template. However, Implementation is not relevant in describing patterns for the high levels of abstraction in agent oriented systems development.

There is no clear rationale for the choice of the view specific fields of the template. Also, the view specific fields added are not structurally arranged. They appear to be a mix of elements for Forces, Solution and Implementation level related issue (Temporal Context).

Our Template: The template is structured based on the features of a good pattern. This maintains clarity and completeness; facilitates communication among different agent research communities and the industry.

Table 2: Listing of Compared Template elements

	Agent Architectural Pattern Template		Agent Interaction Pattern Template	
GoF [9]	**View Oriented [13]**	**Our Template**	**View Oriented [13]**	**Our Template**
Name Intent Motivation Applicability Structure Participants Collaborations Consequences Implementation Known Uses Related Patterns	Name Aliases Problem Forces Entities Dynamics Dependencies Example Implementation Known Uses Consequences See Also *View category fields* Resource Limitations Control Flow Knowledge handling Reasoning capabilities Autonomy User Interaction Temporal Context Decision making	Name Classification Problem Context Forces: *Goal, Autonomy, Social Ability, Environment, Adaptive Behaviour, Intelligence, Decision and action* Solution: *Control, Knowledge Management, Strategy, Interaction Management, Environmental Interface, Structure, Dynamics* Known Uses Resulting context: *Adaptation/Integration* Related Patterns	Name Aliases Problem Forces Dependencies Example Implementation Known Uses Consequences See Also *View category fields* Roles Messages Temporal Ordering	Name Classification Problem Context: *Goal* Forces: *Environment, Adaptive Behaviour, Security, Language Constraint* Solution: *Interaction Strategy, Structure, Dynamics, Ontology* Known Uses Resulting context: *Introduced Entities, Adaptation/Integration* Related Patterns

The rationale for the choice of sub elements is the reflection of the notions of agency in describing forces and the solution that best resolves these forces. There is emphasis on using concepts that are relevant to the level of abstraction applicable to the template. The sub elements introduced under Solution are based on the main components of general agent architecture. Separately defining these sub elements helps to analyze and compare agent architectural patterns. This also helps to connect these patterns with implementation level patterns.

Forces are elaborated so that unresolved forces can be captured in the resulting context segment of the pattern. The Adaptation/Integration sub element is introduced to offer hints on adapting the pattern to specific projects. This also assists with the integration of a pattern into a relevant pattern language.

6. Conclusion and Further work

In this paper, we present a design for agent pattern templates that we believe improve on the description of agent patterns. We described the InteRRaP and Contract Net patterns using two templates built based on our design. The contribution of the template design we presented is two-fold:

1. A template structure that maintains the central theme and the features of a good pattern and defines amendments to the template with sub elements. This will improve agent pattern communication, adaptation and integration.
2. A template that captures the notions of agency as integral part of template element definition. This will enhance the understanding of the agent concepts and facilitate bridging of the gap between conceptual and implementation levels of agent development.

Our further work includes defining templates for other levels of agent system development.

References

[1] J. Muller and M. Pischel. The agent architecture InteRRaP: Concept and application. Technical Report RR-93-26, DFKI Saarbrucken, 1993.

[2] The Patterns Handbook: Techniques, Strategies, and Applications. Rising L. (Ed). SIGS Books November 1998.

[3] D. Chacon, J. McCormick, S. McGrath, and C. Stoneking. Rapid application development using agent itinerary patterns. Technical Report #01-01, Lochheed Martin Advanced Technology Laboratories, March 2000.

[4] Rao, A.S., & Georgeff, M.P. (1991). Modeling rational agents with a BDI-architecture. R.F. J. Allen, and E. Sandewall (Ed.), Second International Conference on Principles of Knowledge Representation and Reasoning (pp.473-484): Morgan Kaufmann.

[5] Yasuyuki Tahara, Akihiko Ohsuga, Shinichi Honiden: Agent System Development Method Based on Agent Patterns. ICSE 1999: 356-367

[6] Sylvain Sauvage, MAS Development: Reusing through Agent Oriented Design Patterns,

Eight World Multi-Conference on Systemics, Cybernetics and Informatics, SCI'04, Orlando, 18-21 July 2004.
[7] Weiss, M., Patterns for Motivating an Agent-Based Approach, *Conceptual Modeling for Novel Application Domains (AOIS@ER)*, LNCS 2814, 229-240, 2003
[8] Yariv Aridor and Danny B. Lange. Agent design patterns: Elements of agent application design. In *Proceedings of the Agents-98*, 1998.
[9] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.
[10] K. Schelfthout, T. Coninx, A. Helleboogh, T. Holvoet, E. Steegmans, and D. Weyns. Agent Implementation Patterns. Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies (Debenham, J. and Henderson-Sellers, B. and Jennings, N. and Odell, J., eds.), pp. 119-130, 2002
[11] Sandra Hayden, Chirstina Carrick and Qiang Yang. Architectural Design Patterns for Multiagent Coordination. In Proceedings of the International Conference on Agent Systems '99. (Agents'99) Seattle, WA, May 1999.
[12] Clinton Heinze. Modelling Intention Recognition for Intelligent Agent Systems. PhD thesis, The University of Melbourne, 2003.
[13] J. Lind. Patterns in agent-oriented software engineering. online paper, 2002.
[14] R. M. Malyankar: A Pattern Template for Intelligent Agent Systems *Workshop on Agent-Based Decision Support for Managing the Internet-Enabled Supply Chain* , Agents '99,
[15] Oluyomi, A., Karunasekera, S., Sterling, L., An Agent Design Pattern Classification Scheme: Capturing the Notions of Agency in Agent Design Patterns, 11th Asia-Pacific Software Engineering Conference, Busan, Korea, December 2004
[16] FIPA. (1998). Foundation for Intelligent Physical Agents. (Web site with information on the FIPA consortium, its proposals and achievements.) http://drogo.cselt.stet.it/fipa/
[17] Lars Kock Jensen, Bent Bruun Kristensen, Yves Demazeau: FLIP: A Platform to Integrate Embodied Agent Technology. IAT 2003: 103-110
[18] K. Fischer, J. P. Müller, and M. Pischel: Unifying Control in a Layered Agent Architecture. In Proc. of the 1st International Conference on Multi-Agent Systems (ICMAS'95), p. 446, AAAI Press The MIT Press 1995.
[19] Michael J. Wooldridge. Multi-agent systems: an introduction. Wiley, Chichester, 2001.
[20] Smith, R.G. (1980). The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. IEEE Transactions on Computers, C-29(12), 1104-1113.
[21] Goldsmith, S.Y. and Interrante, L.D. (1998). An Autonomous Manufacturing Collective for Job Shop Scheduling. In Proceedings of AI & Manufacturing Research Planning Workshop, Albuquerque, NM, The AAAI Press, pp. 69-74.
[22] Parunak, V.D. (1987). Manufacturing Experience with the Contract Net. *Distributed Artificial Intelligence*, Huhns, M.N. ed., Pitman, pp. 285-310.