International Journal on New Computer Architectures and Their Applications (IJNCAA) 1(3): 780-796
The Society of Digital Information and Wireless Communications, 2011 (ISSN: 2220-9085)

Evaluating Complexity of Task Knowledge Patterns through Reusability Assessment

Cheah WaiShiang[1], Edwin Mit[2] and Azman Bujang Masli[3]
Faculty of Computer Science & IT, UNIMAS 94300 Kota Samarahan Sarawak, Malaysia
{c.waishiang@gmail.com[1], edwin@fit.unimas.my[2]}

ABSTRACT

Reusability assessment of patterns is needed to help pattern designers and pattern developers to check whether a pattern is well-designed. Hence, the outcome from the assessment can be used to improve the current patterns and also to reveal the potential of reusing the patterns in software development. This paper presents the reusability assessment of task knowledge patterns through the proposed metrics. This is a continuous effort to evaluate the potential reuse of the proposed task knowledge patterns for multi agent system development. The reusability assessment proposed in this paper further elaborates reusable of pattern by synthesizing how to evaluate the genericity and complexity of a task knowledge pattern (aka. agent patterns) and its similarity to other patterns in tackling a particular problem. The hypothesis is that a pattern is reusable when it is descriptive and expressive. A case study is presented to showcase that the outcome of the assessment can help to improve the effort to design the task knowledge patterns for reuse purposes. Furthermore, the outcome of the assessment allows the pattern developer to communicate their patterns in quantitative manner. The two main contributions of this paper are first, to determine the design quality of agent patterns and secondly, the introduction of a novel designs metrics for agent patterns and the process to assess the potential reuse of task knowledge patterns.

KEYWORDS

Patterns, complexity analysis, agent, metrics, validation.

1 INTRODUCTION

Agent patterns record the experience in engineering agent oriented systems. Agent patterns have aimed to promote an agent based approach to the outside of the agent community [1]. The use of patterns in agent development can reduce the development cost and time [2], promote reuse and reduce the complexity when developing applications [3]. In addition, it allows the novices to rely on expert knowledge and solve the problem in a more systematic and structured way [4] [5].

To support the adoption of agent patterns for agent development, researchers are working on pattern classification and pattern template. The pattern classification supports the ease of accessibility of agent patterns by arranging the collection of agent patterns in a structured manner. The template structure is used to record the agent development experience in a structured manner.

While various patterns have been introduced, the potential usage of agent patterns does not pay much attention in the current literature. One possible reason is that researchers are working on various template forms and introducing

new agent patterns that are suitable with their context.

This paper presents reusability assessment of agent patterns through metrics. We present how to estimate potential reuse of the task knowledge patterns through assessing the quality of the patterns with the proposed design metrics. The proposed metrics aimed to conduct a complexity analysis on task knowledge patterns of a similar kind to determine the level of explicitness of a particular pattern. Assessing the quality of the agent patterns will provide answer to the question, 'How expressiveness of patterns going to be?' From the estimation value obtained by means of a design metric, the pattern developer may look at the early pattern description and conduct further refinement to improve the design of the pattern.

The task knowledge patterns are reusable artifacts that are introduced for agent oriented software development [6]. "What is knowledge? How is knowledge represented?" The notion of knowledge is defined and modelled in CommonKADS [7], a knowledge engineering methodology. In CommonKADS, template knowledge models are introduced and are viewed as design patterns or "knowledge patterns" for tasks [7]. It contains a predefined knowledge that is represented in the form of reusable model sets for developers. Each of the template knowledge models consists of the following pattern elements:

- General characteristics: Description of the features of a task like goals, typical examples, terminology, input and output.
- Default method: Description of the task knowledge by modelling the actions and control structures for the task type through inference structure and task specification, respectively.
- Method variation: Description of the variation of the default method when dealing with a real application.
- Typical domain schema: Description of domain entities that will use for a particular task type.

This paper consists of five sections. Section 2 presents the background study of this research. The effort in assessing the software patterns which form the background knowledge in this work is described. Section 3 presents the reusability assessment of task knowledge patterns through metrics. We elaborate the proposed metrics with a runthrough example in accessing an information finding pattern. Section 4 presents a case study to reveal the potential reuse of the task knowledge patterns of information finding through the proposed metrics. In the case study, we present a quantitative value of the information finding pattern and reveal our analysis result. To further verify our hypothesis that a high complexity or expressiveness of pattern, the more reusable of the pattern is, an empirical study is conducted. The study reveals the important of the expressiveness in driving the reuse of the pattern in MAS development. The paper concludes by Section 5.

2 BACKGROUND

Patterns consist of various pattern elements that explicitly describe the problem, solution and consequences in clear structure. It encourages the developer or designer to communicate the ideas by explicitly presenting the concepts within the pattern [11, 12]. One way to reveal the potential reuse of the patterns is to demonstrate how the patterns support the maintainability of

software development within a controlled experiment. Another way is to assess the design quality of the agent patterns as inspired by Araban and Sajeeve [8].

The reusability assessment of software components is introduced by Saeed Araban at the University of Melbourne [8]. The aim of the research is to estimate the potential reuse of software components. The outcome of this research has lead to the improvement on the effort of designing for reuse purposes among the developers.

In the work [8], object oriented metrics are used to measure the criteria to ease reuse and design with reuse among two object oriented software components. The software components that have been used for the reusability assessment are the Java package and the Eiffel libraries. From the analysis of the result, the author concluded that Java has a better design for reuse due to the minimum number of children as compared to Eiffel.

The object oriented metrics that have been used are weighted method per class (WMF), number of children (NOC), coupling between object (CBO), just to name a few. The summary of the metrics used is as follow:

Weighted Methods per Class, WMF,

$$\mathrm{WMC} = \sum_{i=1}^{n} C_i$$

WMF is also known as the number of methods by Harisson [9]. It is used to calculate the number of methods occurrences within a class.

Number of Attribute, NOA. The number of attributes is referred to as the total number of attributes that are defined within a class.

Response For a Class, RFC. Response for a class is used to measure the method invocation after receiving a message. It calculates the methods that are potentially executed in response to a message received by a class or object. Also, it is used to measure the connection of the potential communication between the classes and methods. The RFC [10] is defined as,

RFC= |RS|, where RS, the response set of the class, is given by $RS = M_i \cup \text{all } j\{R_{ij}\}$

In which, the response set involves the counting of M, the set of all methods in a class, and R_i, the set of methods called by method *i* in the class. Such methods in R are positioned remotely. Li & Henry [10] defined RFC as a coupling measurement as RFC does not only include the method directly involved by a method but also the method called by other methods in other classes.

Coupling between Objects, CBO. Generally, coupling involves identifying the frequency of connections between the classes and types of connections between the classes like interaction coupling and content coupling. CBO is one of the couplings metrics which is used to determine the relationship among classes. Measuring the CBO happens when methods of one class use the methods or attributes of the others. CBO is also known as fan-out within the traditional software metric. Fan-out is defined as element like attribute or method that the class depends on. The CBO is defined below.

CBO = total number of other classes to which it is coupled.

From the literature, OO metrics have been used to evaluate quality characteristics on various artifacts. They

are metrics for accessing the security code [15], metrics for accessing design pattern quality for games [13] and software application, metrics for UML modelling, metrics for evaluating aspect oriented system, metrics for evaluation XML documents [14], metrics for agent systems.

Works have been done to introduce metric suites for agent oriented models. Franch [16] conducts a quantitative analysis of agent oriented models through evaluating dependencies in i*. Weights are assigned to goals and dependencies and a case study on how to calculate the predictability of an i* model is presented. Sarami et al. [17] propose a complexity metric for agent oriented models and conduct an evaluation of the complexity of the message model and Prometheus models. The metrics are the magnitude of graph, diversity of components in a model, the magnitude of table, link density, and TScore. Grossi et al. [18] propose numerical analysis for the organizational structure of MAS. The equations for measuring the specific graph's organizational structure are proposed. They are completeness, univocity, flatness, cover & chain, overlap, and detour.

In this section, we summarized the object oriented metrics that are used for reusability assessment of software components. Although assessing the design quality of agent patterns is not addressed in the current works, it is worth surveying how people assess the quality of the agent system. This will turn into the terminology for our proposed metrics. It is worth exploring how those practices can be used to introduce the reusability assessment of agent patterns. In doing that, we should treat the object oriented metrics discussed as a baseline notion to measure the agent oriented models in the task knowledge patterns as described in the following section. Since we deal with agent models, object oriented metrics are inadequate for measuring them. It has been suggested that the description of a metric should be accompanied by its quality characteristics and the description of the metric's target group. Effort is needed to demonstrate and support the usefulness and significance of the set of proposed metrics through empirical evaluation [9], [19].

3 METRICS FOR TASK KNOWLEDGE PATTERNS

In this section, metrics to measure the complexity of agent models that are used in describing task knowledge are presented. When proposing the metrics, determining the quality characteristic and who will direct to is needed. Metrics' definition must not be ambiguity or over emphasis; facilitate effort in data collecting from the raw data; demonstrating and supporting the usefulness and significance of the set of proposed metric through empirical evaluation [9] [21][22]. In other words, the metrics must pursue with clear goal; theoretical and empirical validation is needed to show the usefulness of the metrics and needs for automatic extraction for data collection in ease of metrics calculation. Also, the validation will demonstrated the metric could be useful to predict external quality characteristics like modifiability, understandability, analyzability and so on [23]. For example, Chowdhury[20] proposes security metric for source code level. The goal of the proposed metrics is used to bring improvement in the code

structure and preventing any attack. One of the metric had been inspired by CK metrics of RFC and adopted it with a suitable security concern in proposing coupling corruption metric. Case study had been conducted to demonstrate the applicability of the metric within two pieces of Eclipse plug-in and observations had been presented.

Taken the suggestion together with evaluation studies, the metrics that are proposed in this research is aimed to conduct the complexity analysis of task knowledge patterns. The complexity reflects the level of expressiveness of the patterns. In the following sections, we first present the metrics through adoption from OO metrics. Two case studies to conduct the analysis of task knowledge patterns are presented next. An analysis of information finding patterns is presented. This will follow by the analysis of information integration pattern. We further conduct the validation of our analysis result through empirical study, as shown in Section 4.

As is described in Section 2, metrics have been adopted in assessing the reusability of agent models. Since we have modelled the patterns by agent oriented models, we can measure the complexity of agent patterns through metrics. The metrics that proposed in this paper serves as one evaluation method of the patterns. The design metrics that are proposed in this paper focus on the models that have been used to describe the task knowledge. In this case, the proposed sets of metrics are dedicated to measure the goal model, role model and domain model for the entire task knowledge pattern.

In our view, the explicitness of a task knowledge pattern is reflected by the complexity of the agent models used for representing the pattern. In order to introduce reusability assessment of task knowledge patterns, we refined the common object oriented metrics like weighted methods per class, size metrics, response for a class, and coupling between objects. The refinement is needed because pure object oriented metrics like the numbers of classes, dependencies between classes, and so on are not adequate for measuring agent models.

Altogether, we propose the following five metrics for complexity analysis: weighted goals per goal model, number of responsibilities, number of domain entities, number of associations per goal, and goal coverage. These design metrics are described below.

Overall goals per goal *(OG)*

Definition: OG = number of goals + quality goals.

We defined the overall goals per goal as the total number of goals that are required to be achieved in order to solve a particular problem. In a goal model, goals, sub-goals and quality goal model an overall achievement of the goal. To fulfil a goal requires fulfilling the sub-goals as well as the quality goal given. The minimum number of *OG* is one. The *OG* includes the count of an initiate goal (e.g. root goal or sub-root goal) as the root goal contributes to the overall achievement of the goals. The *OG* can be measured for a particular sub-goal regardless of whether it is a root goal, sub-goal or quality goal and regardless of the hierarchy and sequence of goal arrangement.

Having a higher value of OG introduces a complexity of the pattern. However, it increases the explicitness of the pattern by explicitly describing the goals that are required to be achieved for task accomplishment. In fact, a higher value

of OG increases the likelihood for reuse. According to Araban & Sajeev [8], software components that are measured with a higher value of weighted method per class, WMC are easier to reuse. The WMC is the number of functionalities in a class. The higher the value of WMC is, the higher the number of functions that are useful for application development also is.

Number of responsibilities *(NoR)*

Definition: NoR = The number of responsibilities.

This metric involves counting the number of responsibilities listed for a particular role in its role model. The higher value of responsibilities may increase the reusability of the pattern as it explicitly lists the subtasks required to be performed in order to achieve the goal(s) that the role is related to.

Number of domain entities (*NoD)*

Definition: NoD = The number of domain entities.

This metric involves counting the number of domain entities within a domain model. Domain entities explicitly describe the knowledge items that are required for fulfilling the responsibilities and achieving the goals directly or indirectly related to them. As a result, the higher the value of NoD is, the more likely the reuse of the knowledge embodied in domain entities also is.

Number of quality goals per goal (*NoQ)*

Definition: NoQ = The number of quality goals that are related to a goal.

Harrison et. al [9] propose the "number of associations per class" metric as an inter-class coupling metric. In our work, the number of associations per goal is redefined as the number of quality goal that are associated with a particular goal, either directly or through its parent goals. A quality goal describes a non-functionality requirement in relation to a goal.

Response for Goal *(RFG)*

Definition: RFG = the number of goals + the number of quality goals + the number of roles for a goal.

Just like the response for class RFC in object oriented metrics, the *RFG* identifies the coverage of a particular goal. The *RFG* metric indicates various aspects of the problem (i.e., subgoals) that have been modelled for a goal together with the associations that support the achieving of the goal. The metric indicates the achieving of the subgoals together with the achievement of the parent goal. The *RFG* metric also includes the relevant quality goals and the roles that are required for achiving the goal. All in all, the value of *RFG* expresses the number of aspects of a problem that have to be considered for achieving the goal.

Figure 1 shows an example of measuring the overall goals per goal for a task knowledge pattern of information finding. In Figure 1, the value of OG for the overall goal model is 11. It includes the sub-goals of organizing result, accepting user request as query, and collecting result. The latter has been further divided into the sub-goals of locating information sources, conducting search, producing relevant result, and displaying result, and the quality goal of high user satisfaction. On the other hand, we can obtain for the subgoal 'Collect result' the value of OG 7. This value of OG is calculated based on the goal of collecting result, sub-goals of locating information sources, conducting search,

producing relevant result, traversing on information sources, and matching information source, and the quality goal of high user satisfaction. The latter applies to the highest-level goal "Manage information finding" and all its subgoals.

The value of NoQ for any goal included by the goal model in Figure 1 is 1. In the figure, achieving the goal 'Manage information finding' is characterized with the quality goal 'High user satisfaction'. This quality goal explicitly represents an extra effort that is required to be considered for achieving the goal 'Manage information finding'. A higher number of NoA increases the likelihood for reuse because the corresponding goal model explicitly describes the additional and "softer" knowledge elements that are required to be considered in solving the problem. From Figure 1, we can calculate that the RFG for the root goal 'Manage information finding' is 14. We can interpret this value so that achieving the goal 'Manage information finding' is responded to by its subgoals and by the roles that relied upon to achieve the given goal. As another example, the RFG for the subgoal 'Collect result' is 11, expressing that fulfilling the goal requires fulfilling the subgoals of locating information sources, conducting search on information sources, and producing relevant result, and the quality goal 'High user satisfaction' and also requires the involvement of the Finder, User, and ResourceManager roles.

4 VALIDATING TASK KNOWLEDGE PATTERNS

Several patterns for the same problem may exist due to the differences of patterns proposed by different people. This research has drafted several examples that model the task knowledge through agent models. Each of the examples models the shared experience through the goal model, role model, organization model, and domain model. The challenges are how to measure the quality of various versions of the patterns and how to show the differences among those patterns. The answer is through reusability assessment. In other words, the quality of patterns can be measured through the proposed metrics.

In this section how a pattern developer can estimate the reuse potential of a task knowledge pattern will be demonstrated. Two case studies on analyzing task knowledge patterns are presented. In the case study I, we present the analysis result of the information finding patterns. On the other hands, the case study II presents the analysis result of the information integration pattern.

4.1 Case Study I: Analysis of Information Finding Patterns

In the following description, analysis of information finding patterns is presented. A detailed elaboration on the information finding patterns is presented in [24] and [25].

To assess the pattern, the estimation value can be obtained through adopting the metrics presented in section 3. They are overall goals per goal (*OG*), a response for goal (*RFG*), number of quality goals (*NoQ*), number of responsibilities (*NoR*), and number of domains (*NoD*). We can pinpoint the reuse potential of a task knowledge pattern by first indicating that the pattern consists of a certain number of goals, response for goal, number of responsibilities, number of domains, and number of associations. Then, we can estimate the reuse potential based on

those values. In the following description, we present the reusability assessment of the task knowledge pattern that is modelled in [24]. This is followed by the reusability assessment of eight task knowledge patterns of information finding as presented in [25]. Further elaboration of the estimation values is given at the end of this section. The values have been calculated as follows:

The *NoR*

= Number of responsibilities$_{informationFinding}$
= Number of responsibilities for the role of InformationFinder +
Number of responsibilities for the role of QueryFormulator
= 8 + 8
= 16

The *NoD*

= Number of domains(D)$_{informationFinding}$
= D1:Query + D2:RelevantContent + D3:Information Sources +
D4:Criteria+D5:Domain + D6:User
= 6

The *RFG*

= Number of goals(G)$_{informationFinding}$ + Quality goal(QG)$_{informationFinding}$ + Role(R)$_{informationFinding}$
= G1:Manage Info finding + G2:Produce query + G3:Analyze query language +G4:Matching rule interpretation + G5:Handle query expansion
+ G6:Handle domain + G7:Conductinterpretation
+ G8:Organize result + G9:Obtain request
+ G10:Awaiting query + G11:Obtain result
+ G12:Obtain relevant references +G13:Conduct search + G14:Traversing & extract content
+ G15:Matching + G16:Display result
+ QG: High user satisfaction + R1: Finder
+ R2:Initiator + R3:QueryFormulator
= 16G+ 1QG + 3R
= 20

The *OG*

= Number of goals(G)$_{information\ Finding}$
+ Quality goal(QG)$_{informationFinding}$
= G1:Manage Info finding + G2:Produce query
+ G3:Analyze queryLanguage
+ G4:Matching rule interpretation + G5:Handle query expansion
+ G6:Handle domain
+ G7:Conduct interpretation + G8:Organize result + G9:Obtain request
+ G10:Awaiting query + G11:Obtain result
+ G12:Obtain relevant references
+G13:Conduct search + G14:Traversing &extract content + G15:Matching
+ G16:Display result
+ QG:High user Satisfaction
= 16G + 1QG
= 17

The *NoQ*

= Number of quality goals $_{informationFinding}$
= QG:High user satisfaction
= 1

The number of responsibilities for the task type (NoR) is 16. The number of domain entities (NoD), which have been explicitly modeled in the pattern is 6. The overall goals per goal (OG) is 17. This value expresses that expanding or reformulating a user query to increase the number of relevant results returned may be required for information finding. In addition, the user should be allowed to provide his/her preferences other than the solution given as well as the returned documents should be arranged accordingly. The response for the root goal (RFG) is 20. Two roles have been shown to be important when conducting the task of information finding. These roles are managing the finding, which involve handling of queries, conducting search, ranking, and combining results, and supporting query interpretation and expansion. The number of quality goals (NoQ) for this type of taskis 1. Achieving the quality of the goal; user satisfaction is important in information finding. Hence, the solution must be able to return a collection of relevant results either according to the user preferences or within a certain degree of relevance. For example, when performing a query, efficiency of the retrieval should be considered. In such a case, the time

required for returning the results becomes an aspect to the solution.

Table 1 presents the metrics that characterize the task knowledge patterns of information finding: *if1, if2, if3, if4, if5, if6, if7, if8* and *TKP*. This is the outcome based on the reusability assessment through the proposed design metrics.

Once the result is obtained, the explicitness and comprehensiveness of the patterns can be determined through the following guidelines.

1. A pattern is claimed to be explicit and comprehensive if it has the best score on each of the metrics listed in the Section 3.
2. An agent will play a role and serve its responsibilities towards achieving the goal(s). If the pattern has explicitly described the goals and responsibilities in detail, the pattern is claimed to be explicit and comprehensive.
3. A pattern that scores well in RFG but scores low for NoA, NoR and NoD as compared to others pattern is claimed to be explicit due to the reason that having a higher number of RFG has indicated the explicitness of the goals and the person in charge (i.e. role) in achieving the goals. This is important as goals and role are important elements for agent paradigm as mentioned in the previous guideline.

Presented in Table 1, the highest values of the metrics OG, RFG, NoD, NoQ, and NoR characterize the pattern modeled in Figure 2 of [24]. This finding indicates the explicitness of the task knowledge pattern (TKP). This is a comprehensive task knowledge pattern because it has been derived from various articles. This confirms our initial assumption that the TKP pattern is more reusable as compared to the others because it takes into consideration more sub-goals and other elements. This finding complies with the claim that having a higher number of methods in an OO class leads to more reusability of the corresponding software component [8]. A further observation from the results presented in Table 1 is that the next pattern in terms of comprehensiveness and explicitness is the *if3*. This is because *if3* pattern has scored slightly less number of goals and response for goal as compared to TKP. Other than that, the pattern of *if3* has scored well as compared to the others.

The three remaining groups of people, if1, if7 and if8 have produced slightly less comprehensive task knowledge patterns. These patterns are *if1*, *if7* and if8 accordingly. Finally, the level of explicitness for the rest of the patterns can be arranged accordingly: the pattern that described at *if2* to the pattern that described from *if5*, *if6* and *if4*. Adopted from the guideline 3, the RFG for *if5* is higher than *if4* and *if6* although *if6* scored 10 in NoR. As a result, we claim that the pattern of *if5* is more explicit as compared to *if6*.

In this section, estimating potential reuse of the task knowledge patterns is explained. Based on the estimation value, we may improved our task knowledge pattern (TKP) that is modeled in Figure 2 of [24] with an additional quality goal (appropriate manner), additional responsibilities (e.g., monitoring and recording troubleshooting cases) and additional domain entities (e.g. error) that have been derived from the patterns *if1*, *if2*, and *if3* modeled. The improvement is needed to reduce the level of explicitness on a particular element within the pattern. In addition, the improvement is required to make our TKP pattern more comprehensive and explicit which we

believe will lead to better reusability of the pattern.

4.2 Case Study II: Analysis of Information Integration Pattern

Result of the reusability assessment for the information integration pattern that presented in Appendix C is shown in this section. The pattern is the best scores among the others patterns for the similar problem. The values show the measurements of the overall goal model, role model, and domain model of the TKP pattern. The values have been calculated as follows:

The *NoR*
= Number of responsibilities$_{integration}$
= Number of responsibilities for the role of Inter-operator
+ Number of responsibilities for the role of AccessController
+ Number of responsibilities for the role of User
+ Number of responsibilities for the role of ResourceManager
= 5 + 1 + 2 +1
= 9

The *NoD*
= Number of domains(D)$_{integration}$
= D1:Task + D2:Service
+ D3:Resource + D4:User+D5:Constraint
= 5

The *RFG*
= Number of goals(G)$_{integration}$
+ Quality goal(QG)$_{integration}$ + Role(R)$_{integration}$
= G1:RegisterService... + G2:ReceiveService
+ G3:ReformulateTask...
+ G4:HandleTransaction...+ G5:GenerateResult
+ QG:Secure
+ R1:Client/User +R2:AccessController
+ R3:Inter-operator +R4:ResourceManager
= 19G+ 1QG + 4R
= 24

The *OG*
= Number of goals(G)$_{integration}$
+ Quality goal(QG)$_{integration}$
= = G1:RegisterService... + G2:ReceiveService
+ G3:ReformulateTask...
+ G4:HandleTransaction...+ G5:GenerateResult
+ QG:Secure
= 19G +1QG
= 20

The *NoQ*
= Number of quality goals $_{integration}$
= QG:secure
= 1

5 EMPIRICAL EVALUATION OF INFORMATION INTEGRATION PATTERN

Section 4.1 and 4.2 present the results of the patterns in a quantitative manner. We present the best scores among the information patterns and integration patterns. In other words, the TKP of [24] and pattern in Appendix C are comprehensive and expressiveness as compared with the other patterns. Indirectly, those patterns are reusable. To further validate the analysis result, an empirical study is conducted.

The empirical study is needed to validate the usefulness of metrics in determining the quality attributes of reusability.

A questionnaire was prepared for conducting the study. The questionnaire was designed to assess the content of the patterns that had been expressed by agent-oriented models and assess the learnability and usefulness of the patterns. A survey was conducted with two Master's students at Tallinn University of Technology, Estonia, who were both novices in agent-oriented software development. Those students were respectively required to develop an agent-oriented recommendation system and an agent-oriented interoperability system for their Masters Theses projects. At the beginning of completing these projects, the students explored agent-oriented modelling. After that, the students were presented with task knowledge patterns for agent-oriented development. One of the students is

working on information integration pattern. The other is working on recommendation pattern as described in [25]. They were required to study the patterns before they started to design agent-based systems for their respective application areas. The students had approximately two months for designing agent-oriented software systems facilitated by task knowledge patters. Upon completion of the project, the students were provided with questionnaires to evaluate the task knowledge patterns that were adopted by them. The answer for the questionnaire form that was employed is listed below.

Question: *Does the pattern provide sufficient information for you to accomplish a task given or solve a particular problem? Why?*
Just about right. I always like when the pattern includes a sample implementation of itself in the real life context (such as how it could be implemented/used/applied in a concrete problem solving). This was missing from the pattern description and would be valuable addition. It helps to better comprehend how the pattern could be implemented / used.

Question: *Is the pattern given able to work across various application domains?*
Yes. The pattern is not related to problem domain, its related to how to solve a certain system requirement such as interoperability between different parties and resources.

Question: *How useful do you find the emerging task knowledge patterns in agent oriented software development?*
Easy

Question: *Is it preferable for you to refer to task knowledge patterns prior to agent oriented software development or do you prefers to use another method?*
I suppose it depends on the requirements and problem at hand. With agent oriented system, I'm definitely more inclined to use the patterns meant for agent oriented software development.
The results of the survey are described in the previous description. We can interpret the results in the following way. In general, novice users (e.g., students) seem to be satisfied with the usage of task knowledge patterns that have been expressively described. The students agree that the task knowledge patterns were useful when developing multi-agent systems and were easy to learn. According to the surveys, the patterns were easily able to communicate ideas and concepts behind task knowledge patterns and both students preferred to adopt the patterns also for future multi-agent system development. In other words, task knowledge patterns facilitated solving the problem at hand for both students. Consequently, our hypothesis on the higher level of expressiveness able to introduce the reusable of agent pattern is valid.

6 CONCLUSIONS

A novel reusability assessment method for task knowledge patterns is introduced in this paper to estimate the quality of the patterns. Estimating the potential reuse of the patterns is needed to help the pattern designer or pattern developer to check if the pattern is well-designed. The reusability assessment proposed in this paper further elaborates reusability assessment by synthesizing how to evaluate the complexity and

hence after expressiveness of a task knowledge pattern and its similarity to other patterns in tackling a particular problem. Several design metrics are introduced to measure the complexity of the task knowledge patterns. With the help of the metrics, the values received from measurements can be used to improve the patterns. As a continuation from this work, we are working on extending our assessment method to others agent patterns. As mentioned before, knowing the design quality of agent patterns is important. In fact, we believe that by understanding the potential reuse of the patterns will better improve the adoption of agent technology to wider software practitioners, in which there is much more to explore in future.

7 REFERENCES

1. Weiss, M. A pattern language for motivating the use of agents. 5th International Bi-Conference Workshop of Agent oriented Information System. Melbourne, Australia: 142-157 (2004).
2. Jureta, I., M. Kolp, et al. Patterns for Agent Oriented e-Bidding Practices. 9th International Conference of Knowledge-Based Intelligent Information and Engineering Systems. Melbourne, Australia. 3682: 814-820 (2005).
3. Lima, E. F. A., P. D. L. Machado, et al. (2004). "An approach to modelling and applying mobile agent design patterns." ACM SIGSOFT Software Engineering Notes 29(3): 1-8 (2004).
4. Mouratidis, H., P. Giorgini, et al. Security patterns for agent systems. Proceedings of the Eighth European Conference on Pattern Languages of Programs (EuroPLoP), Wiley, New York (2003).
5. De Wolf, T. and T. Holvoet. A catalogue of decentralised coordination mechanisms for designing self-organising emergent applications. Technical Report, Department of Computer Science, K.U. Leuven (2006).
6. WaiShiang C., L. Sterling, Taverter K.. Task knowledge patterns reuse in multi-agent system development. Proceedings of the 13th International Conference on Principles and Practice of Multi-Agent Systems, Kolkata, India, November (2010).
7. Schreiber, G. Knowledge engineering and management: the CommonKADS methodology, MIT press (2000).
8. Araban, S. and A. S. M. Sajeev. Reusability Analysis of Four Standard Object-Oriented Class Libraries. International Conference on Software Engineering Research, Management and Applications, SERA 3647: 171-186 (2006).
9. Harrison, R., S. Counsell, et al. An overview of object-oriented design metrics. Proceedings on Eighth IEEE International Workshop on incorporating Computer Aided Software Engineering: 230-234 (1997).
10. Li, W. and S. Henry. Object-oriented metrics that predict maintainability. Journal of systems and software 23(2): 111-122 (1993).
11. Beck, K., R. Crocker, et al. Industrial experience with design patterns. Proceedings of the 18th international conference on Software engineering, IEEE Computer Society: 103 – 114 (1996).
12. Chung, E. S., J. I. Hong, et al. Development and evaluation of emerging design patterns for ubiquitous computing. Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques ACM New York, NY, USA: 233-242 (2004).
13. Cutumisu, M., et al. Evaluating pattern catalogs: the computer games experience. in Proceedings of the 28th international conference on Software engineering: ACM New York, NY, USA (2006).
14. Klettke, M., L. Schneider, and A. Heuer. Metrics for XML document collections. in XMLDM Workshop, Czech Republic. (2002).
15. Chowdhury, I., B. Chan, and M. Zulkernine. Security metrics for source code structures. in Proceedings of the fourth international workshop on Software engineering for secure systems: ACM New York, NY, USA (2008).
16. Franch, X. On the quantitative analysis of agent-oriented models. 18th International Conference Advanced Information Systems Engineering. Luxembourg. 4001: 495-509 (2006).
17. Saremi, A., M. Esmaeili, et al. Evaluation complexity problem in agent based software

development methodology. International Conference on Industrial and Information Systems, ICIIS (2007).
18. Grossi, D., F. Dignum, et al. Structural aspects of the evaluation of agent organizations. International Workshop of Coordination, Organizations, Institutions, and Norms in Agent Systems. **4386:** 3-18 (2007).
19. Genero, M., M. Piattini-Velthuis, et al. (2004). Metrics for UML models. UML and Model Engineering 5(1): 43.12.
20. Chowdhury, I., B. Chan, and M. Zulkernine. Security metrics for source code structures. in Proceedings of the fourth international workshop on Software engineering for secure systems: ACM New York, NY, USA (2008).
21. Genero, M., et al., *Metrics for UML Models.* UML and Model Engineering, **5**(1): p. 43 (2004).
22. Genero, M., D. Miranda, and M. Piattini. *Defining and validating metrics for UML statechart diagrams*. in *In 6th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering*.
23. Yi, T., F. Wu, and C. Gan. *A comparison of metrics for UML class diagrams*. in *ACM SIGSOFT Software Engineering Notes*. (2005).
24. WaiShiang C., Mit E., Reusability assessment of task knowledge patterns through metrics, The 2nd International conference on software engineering and computer system, (ICSECS 2011) 27-29 Jun 2011, Universiti Malaysia Pahang, Kuantan, Malaysia (2011).
25. WaiShiang C. (2010). Patterns for Agent oriented software development, The Melbourne University. PhD.

APPENDIX A

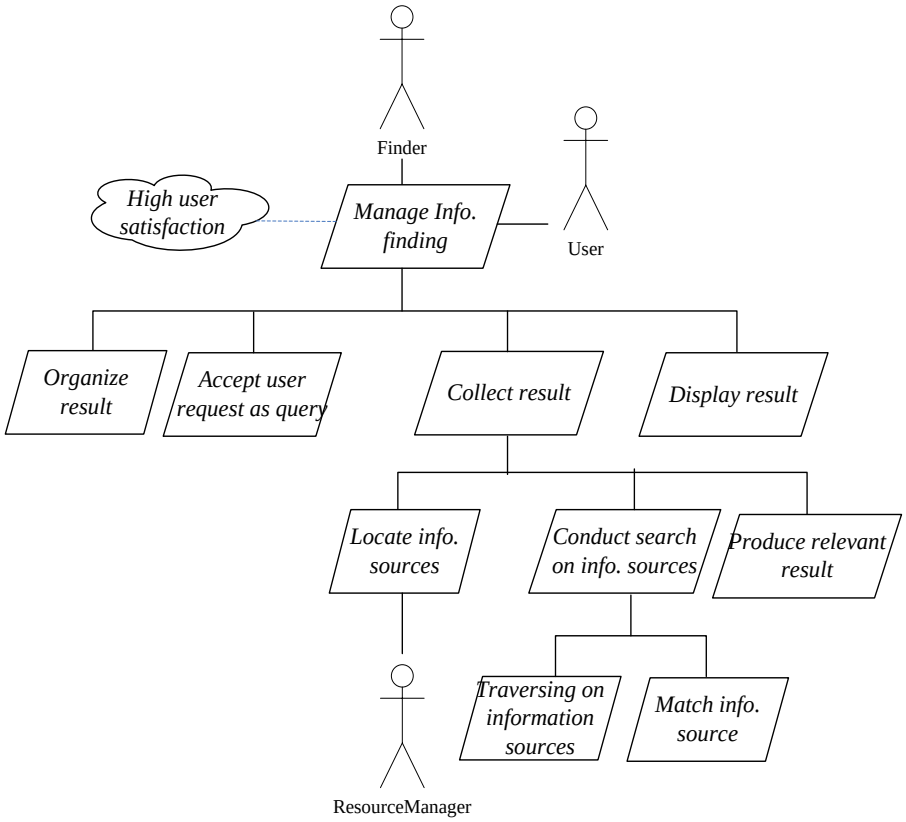


Figure 1: OG_{root} = 11, $OG_{'organizeQuery'}$ = 7 for the task type of 'information finding'

APPENDIX B

Pattern	***OG***	***RFG***	***NoQ***	***NoR***	***NoD***
TKP	17	20	1	16	6
if1	10	12	3	12	7
if2	4	6	1	8	4
if3	12	16	0	19	5
if4	5	7	0	5	4
if5	6	8	0	5	2
if6	4	7	0	10	4
if7	8	10	0	12	3
if8	8	11	1	13	4

Table 1: The values of metrics for the task knowledge patterns of information finding

APPENDIX C:

Information Integration pattern

Intent:
The purpose of this task type is to handle the accessibility of information across various information sources.

Also known as:
MOMIS, IBHIS

Context / Applicability:
Use this pattern when
-you want to perform operation (e.g. execution of task or sending a query) across the environment ranging from diverse operating system, programming system [in1], platform, resources, vendor related (e.g. various travel agent system, various payment system, various hotel management system, car rental system and so on.

Problem: Dealing with task type to integrate or support accessibility among people and computer systems in large, geographically resources.

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.

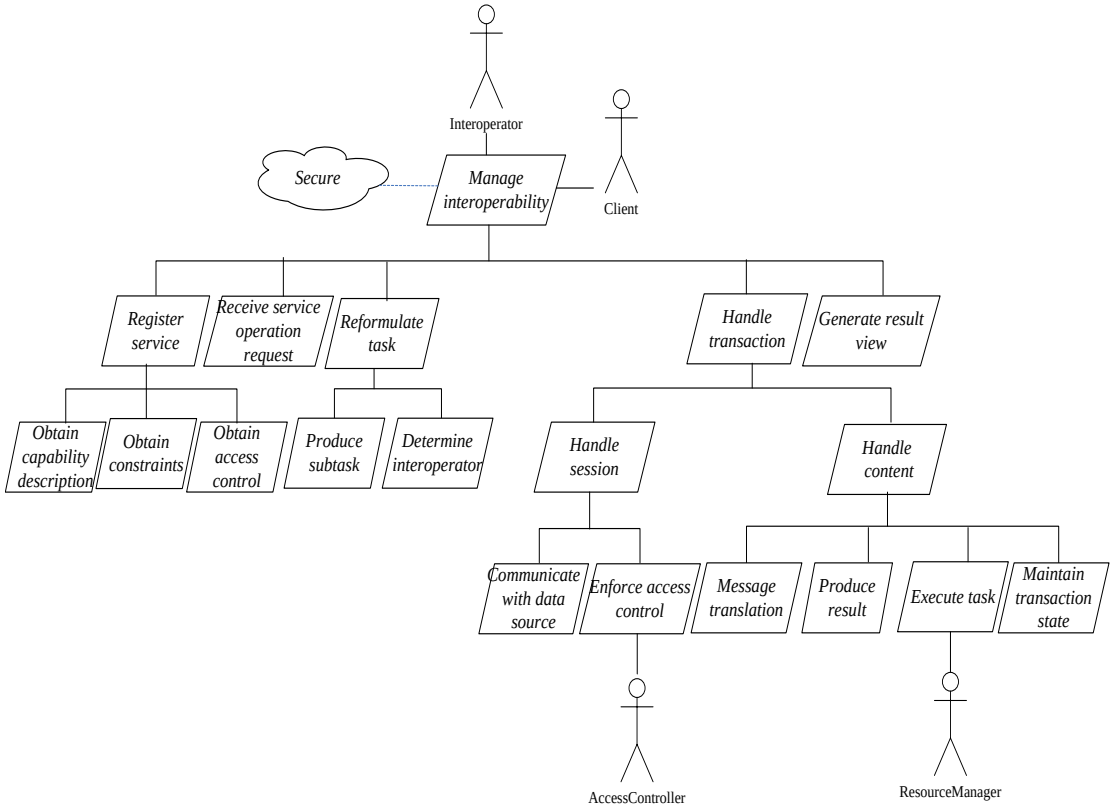
Goal: The solution given must focus on the accessibility issue in handling the interoperability. Normally, the system will range from various resources as well as processing involved in task execution. In this case, a task will require to perform certain operation on other platforms, applicationor various resources form. User query must able to communicate across a wide range of services transparently. The solution given will support query handling easily regardless of platform or vendor specify instruction from one task to the other. The solution given will maintain the availability or activation of service provided.

Quality goal: Ensure the accessibility does not breaking through any protection given.

Role: Four roles have been designed in dealing with this task type. They are role played as client that required information or perform certain operation, resourceManager to provide the information or preparing the execution environment and role like interoperator and accessController to control and arrange the accessibility level.

Resource: This task type requires domain entities of task, constraint, user, service and resource.

Solution:



Figure< **Goal Model**>. Goal model for task type of interoperability

Table <**Role model**>**.** Role model for task type of interoperability

Role name	Inter-operator
Description	Support accessibility across wide range of application, vendor specify application.
Responsibilities	setting up the interoperability environment - support service registration by waiting on incoming service registration or updating. -problem reformulation working on the interoperability operation -handling proxy -handling content layer: resource handling and operation handling by casting the application according to the service registration form; query translation and result transformation. -generate diverse view or result
Constraints	-The service must exist.

Role name	AccessController
Description	Handle access control on the vendor specify function or module to prevent the violation of the system operation within an application.
Responsibilities	Enforce access control according to service requested.
Constraints	-Service registration must provide with access right if needed.

Role name	User/Owner/Initiator

Description	Handle service request and response.
Responsibilities	-Handle service to do by sending service request. -Obtain response.
Constraints	- The received service will direct to interoperator. - Provide authentication identification if needed.

Role name	ResourceManager
Description	Platform for resource execution (e.g. perform searching or tool for data analysis [in2])
Responsibilities	-Execute task within given context
Constraints	-Perform task execution according to specified constraints (e.g. authentication and authorization)

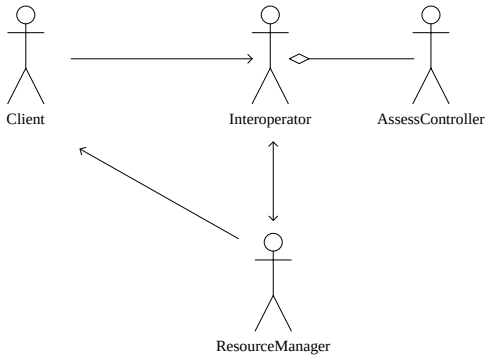


Figure <Organizational model>. Organization model for task type of interoperability

Organization structure The interaction among the client and interoperator happens that the interoperator will ensure the accessibility of information across various resources once received the user request. To prevent the violate that occurred during the transaction, interoperator will interact with AssessController. The ResourceManager will conduct execution (e.g. retrieve information, running computation upon the interaction with interoperator and produce the result to client.

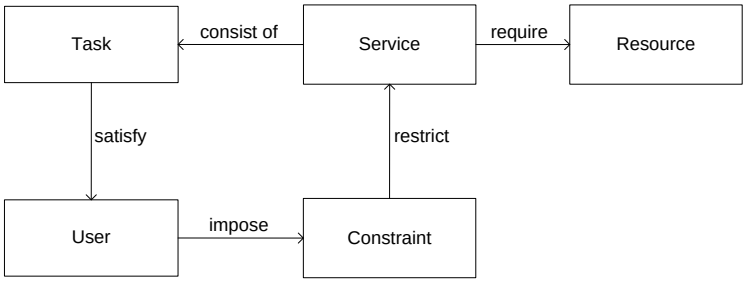


Figure <Domain model>. Domain model for task type of interoperability

Consequences:

The benefit from integration is to provide user upon processing across resources and application;

-reduce the risk to avoid unaware or incorrect interpretation of patient history reside within various resources and platform.

-Support the current standalone information system without further changes.

-Presenting the result in uniform view instead of multiple interfaces that occur or represented across various system.