Enhancement of Rapid Object Process for Embedded System (ROPES) Metamodel To Support Pattern-Oriented Development

Mohd Adham Isa
Software Engineering Department
Faculty of Computer Science and Information System
Universiti Teknologi Malaysia
81310 Skudai, Johor.
mohdadham@utm.my

Dayang Norhayati Abang Jawawi
Software Engineering Department
Faculty of Computer Science and Information System
Universiti Teknologi Malaysia
81310 Skudai, Johor.
dayang@utm.my

ABSTRACT
Embedded Real Time (ERT) software has becoming more complex due to real time requirements and resource constraints. ERT system tries to persuade the combination of hardware and software but at the same time, ERT system properties such as timing correctness and response time must be adopted. The complexity of ERT system properties perceived up the complexity of ERT software development. Applying patterns to ERT system development process poses the systematic software analysis and design. Patterns are used systematically in each phase to wrap system structure and behavior in pattern shapes. ROPES is the software development that specific for ERT system and focuses at analysis, design, implementation and testing phase. ROPES were using patterns only at design phase and do not explicitly address the uses of patterns from analysis phase. The ultimate aim of this paper is to propose a Metamodel for integrated Pattern-oriented Analysis and Design (POAD) and Rapid Object Process for Embedded System (ROPES) to inspire the uses of patterns in ERT system development. The principle aim for this Metamodel is to construct the model that will provide the subtle description for specific ERT system requirements. The Metamodel also shows the structure of ERT system model that exhibit the highly uses of patterns.

Categories and Subject Descriptors
Software Engineering- Software Analysis, Software Design, Requirements Engineering

General Terms
Metamodel, systematic

Keywords
Pattern, Embedded Real-time (ERT)

1. INTRODUCTION
Embedded real time (ERT) is a system that contains hardware and software and highly depending on the correctness of the time execution, response time and scheduling. Most of the embedded systems are characterized as real time systems, which consist of real time properties such as response time and worse case execution time [1].

Nowadays, the trend which is developing systems from the scratch eventually causes the problems like failure to meet customer requirements, budget constraint and extend deadline. The same problems faced by ERT system due to its sophisticated properties. The complexity of ERT system properties perceived up the complexity of ERT software development. ERT software developments are crucial since the combination both hardware and software can lead into the complexity of the process. All of this events must have a systematic process in order to reduce the time and cost of development. Thus, implementing ERT system from scratch does not guarantee the productivity and quality. Then, in order the overcome those problems, it was realized that the software reusability is the essential factors that contributed to minimize the existing problems [2, 3, 4, 5]. Software reusability allows parts of software component to be renewed or replaced by existing components. Thus, software reusability helps us to concentrate on adding more complex system functionalities rather than focusing on developing basic components.

A pattern was introduced into the software development process as a means of manipulating hard-earned experience from the previous problems [6]. [7] define a pattern as a software designs problems that frequently occurs in software development and then provides the explicit solution that it can be reused. The advantages of using patterns are [6]. 1) the reuse experienced, 2) the development of standards types of solution, 3)a normalized mechanism for abstracting form multiple example and method for categorizing problems. Thus, software pattern is a software reuse approaches and provides a way to reuse establish solution for specific problems that can be applied across domain at any level of development. Hence, software pattern like analysis [8], design [9, 10, 11] and programming patterns [12] is becoming popular and widely used among the developers and researchers. Although most of the common use of pattern at the design phases [9, 10, 11], many researchers claims and believes that the use of pattern at analysis phases could emphasizing the higher degree of reusability and software quality [14].
POAD is the software development methodology that non-specific domain and focuses at analysis and design software development phase. POAD uses patterns as main features in its development. ROPES is the software development that specific for ERT system and focuses at analysis, design, implementation and testing phase. ROPES were using patterns only at design phase and do not explicitly address the uses of patterns from analysis phase. In order to enable high software reusability on ROPES, POAD methodology is combined into ROPES methodology so that patterns are systematically used in ROPES methodology. Therefore, patterns will allow parts of software components to be renewed or replace by existing patterns in patterns library.

The aim of this paper is to develop integrated Pattern-oriented Analysis and Design (POAD) [7] and Rapid Object Process for embedded System (ROPES) [13] metamodel. ERT system is modeled from the initial process in analysis phase and realizes the outcome into the design phase. All of this process is presented in pattern shapes which are describing the system structural and behavior. Moreover, the integrated POAD and ROPES metamodel also enable a systematic reuse through pattern and to support pattern-oriented development for ERT system.

This paper is organized as follows. In Section 2, the concept of software patterns are discussed. Section 3 describes the development of POAD and ROPES metamodel and how two metamodels are integrated together to enable the high reuse of patterns in ROPES methodology. Finally, the discussion and conclusion of this paper are presented in Section 4 and 5 respectively.

2. INTEGRATION METHODOLOGY

POAD is the software development methodology that non-specific domain and focuses at analysis and design software development phase. Thus, ROPES is the software development that specific for ERT system and focuses at analysis, design, implementation and testing phase and were using patterns only at design phase. However, to enable the integration of both model, there are things need to be considered. The two things are integration process and criteria of integration process.

Integration process is the process of integrate POAD and ROPES methodology to produce integrated metamodel that support the use of patterns in software development process. Figure 1 shows the integration process of POAD and ROPES methodology. The integration process firstly defined the POAD and ROPES metamodel syntax and semantics. Then, both metamodels are mapped together to discover the differences and similarity between both metamodel. However, there are three main criterias need to be considered during the metamodel mapping process. Those criteria are 1) POAD exclusive, 2) ROPES exclusive and 3) Share exclusive. POAD and ROPES exclusive means the metamodel is exclusively belong to each metamodel. Share exclusive means both metamodel shares the same UML metamodel definition. Then, metamodel process takes part as final stage of integration process. Metamodel process is the process of embedding POAD development process into ROPES development process in order to enable the use of patterns in ROPES software development process.

In conclusion, the integration methodology has two main processes which are integration process and criteria of integration process. Integration process is the process of integrating POAD and ROPES methodology. This process have involves a few steps and constraints. The criteria that need to be considered during integration process are metamodel mapping and software development process mapping. Those criteria are essential in order to enable the integration both metamodels running smoothly and in a correct order.

![Figure 1. Integration Process](image)

3. THE PROPOSED METAMODEL

In this section, we present the construction of POAD and ROPES metamodel based on description on its development process. As indicated before, the metamodel describes the structural representation model based on its requirement.

Moreover, after the POAD and ROPES metamodel are defined, the integrated POAD and ROPES metamodel will be defined by going through mapping process. Mapping process for both methodologies is crucial to identify the similarity and differences of metamodel elements that contains in their metamodel taxonomy. After the comparison results are identified, then the integrated POAD and ROPES metamodel is defined in order to facilitate the uses of patterns in ROPES development process. Moreover, the integrated metamodel also introduces the taxonomy of its metamodel from analysis until design phase.

3.1 POAD Metamodel

The first elements that need to be defined is a metamodel for POAD methodology. POAD methodology is a general approach of software development which is not domain specific and it defines the composition of patterns at high level by using UML as
a means to captures all the software requirements. UML is the collection of Meta model notation offers by OMG and consists of high level Meta model abstraction. In order to define the POAD Meta model, firstly we must captures all the artefacts produced in each development process and analyse the relationship between artefacts and UML Meta model. Then, we identify the respective Meta model and construct the POAD Meta Model. Table 1 illustrates the artefacts produced in each phases

<table>
<thead>
<tr>
<th>Phase</th>
<th>Artefacts produced</th>
<th>UML Meta model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>A set of pattern with name</td>
<td>Namespace (from Core)</td>
</tr>
<tr>
<td>Design</td>
<td>Patterns with interface</td>
<td>Relationship</td>
</tr>
<tr>
<td>Design Refinement</td>
<td>Class diagram Pattern dependencies Pattern association</td>
<td>Class diagram Dependency Association Classifier</td>
</tr>
</tbody>
</table>

Table 1 shows the artefacts produced in each POAD development phase and their relationship with UML Meta model. In analysis phase, a set of patterns are retrieved from the pattern library. Then, the selected pattern will be integrated by using association relationship in UML Meta model.

After those patterns were integrated, we defined the pattern interface in order to enable interaction between patterns. Furthermore, the patterns must define a details class diagram in the pattern itself to cope with the system requirements. Moreover, five key elements originally extended from the UML meta-model to enable pattern-oriented composition using POAD. Those five key elements are:

- Pattern. A set of collaborating classes that originally retrieve from Namespace (from Core)
- PatternInterface. A set of classes and operations that define how the pattern interfaces with the other pattern. Originally retrieved from Classifier.
- PatternDependency. It shows the relationship between patterns.
- PatternAssociation. It shows the relationship between pattern interfaces.
- InterfaceBinder. It used to connect PatternInterface with its internal classes.

However, for real time system, the structural elements alone is not enough. It must cover both structural and real time behaviour elements in other to represent how the ERT system should react with the time constraints and resources. [15] was introduced the enhancement of POAD Meta model that enable explicit description for real time behaviour of components. The two elements that have been introduced [15] are 1) RealTimeBehaviour and 2) PatternControlInterface. Figure 2 shows the enhancement of POAD metamodel proposed by [15]. PatternControlInterface and RealtimeBehavior were added into the original POAD metamodel in order to describe the real-time behavior requirements.

### 3.2 ROPES Metamodel

ROPES methodology defines the systematic development process specific for ERT system. It consists four development phases which are analysis, design, translating and testing and each phase have own activities. ROPES model covers both structural and behaviour elements that represent the whole system views and can be represented in UML metamodel. Therefore, we must systematically identify the artefacts produced for each development phase in order to define the relationship between

![Figure 2. Enhancement of POAD Metamodel [15]](image-url)
artefacts in ROPES and UML metamodel. Table 2 shows the artefacts produced in ROPES methodology.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Artefacts</th>
<th>UML Meta model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>Use Case, Object,</td>
<td>Use case, Object,</td>
</tr>
<tr>
<td></td>
<td>Messages, Actor.</td>
<td>message, actor</td>
</tr>
<tr>
<td>Requirement</td>
<td>Subsystem, component,</td>
<td>Subsystem, component,</td>
</tr>
<tr>
<td></td>
<td>statechart, activity,</td>
<td>port, interface,</td>
</tr>
<tr>
<td></td>
<td>interaction diagram</td>
<td>behaviour,</td>
</tr>
<tr>
<td>System</td>
<td>Class diagram, Object,</td>
<td>Subsystem, component,</td>
</tr>
<tr>
<td></td>
<td>Collaboration, Statechart,</td>
<td>port, interface,</td>
</tr>
<tr>
<td></td>
<td>Activity, Interaction</td>
<td>behaviour,</td>
</tr>
<tr>
<td>Object</td>
<td></td>
<td>componentRealization,</td>
</tr>
<tr>
<td>Design</td>
<td></td>
<td>Dependency, Relation,</td>
</tr>
<tr>
<td>Architectural</td>
<td>Statemachine, Use case,</td>
<td>Association</td>
</tr>
<tr>
<td></td>
<td>Activities, interaction,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Class diagram, component.</td>
<td></td>
</tr>
<tr>
<td>Mechanistic</td>
<td>Class diagram</td>
<td>Pattern, Class diagram</td>
</tr>
<tr>
<td>Detailed</td>
<td>Class diagram</td>
<td>Class diagram</td>
</tr>
<tr>
<td>design</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2 shows the artefacts produced in ROPES development phase and correspond with the UML Meta model. Generally, ROPES are using structure and behaviour Meta model and combining with Real Time Meta model such as Message, Thread and Resources. This elements show that ROPES methodology covers real time system properties in their modelling systems. After we defined the respective UML Meta model for each artefact, then the ROPES Meta model is constructed as shown in Figure 3.

3.3 Integrated POAD and ROPES Metamodel

Mapping POAD metamodel to ROPES metamodel is essential process to discover the similarity and differences between them in order to define the integrated POAD and ROPES metamodel. Direct migration of two methodologies is possible due to the similarity in software development definition. POAD metamodel consists seven primary elements as mention in section 3.1 while ROPES metamodel contains a structural and behaviour UML metamodel elements.

In POAD, Pattern is a set of collaborating classes that solves a specific design problem and cannot inherit internal parts from other patterns. In contrary, Pattern is not Package or Subsystem. Package is a collection of ModelElements and can be parents or children in an inheritance tree. Hence, Package can import or export its internal elements to other Package. However, in POAD, Package is used in modelling the patterns with some limitation and constraint. The limitation is relationship and interfaces. The relationship between patterns in POAD is dependency relationship that is semantically a uses relationship and is not import or export relationship. Thus, interfaces between patterns could be the classes itself or operations on interfaces classes. Then, subsystem is generalization of Package and cannot be considered as patterns.

In addition, there are three types of interfaces in POAD which are PatternInterface, PatternControlInterface and InterfaceBinder. All these elements are defined specifically for Pattern metamodel. They have own definition and purposes. Then, its definition is different from interface elements in ROPES. Interface in ROPES defines the set of operation or classes that enable two subsystem components interface each other. Thus, subsystem and component is physical view of the system rather than logical view. Then, the definition of POAD interfaces is has different semantics meaning from ROPES interface definition.

However, both metamodel consists of similar elements as shown in Table 3 which are Relationship, Constraint and Real-time behaviour. Relationship defines the association or dependency between classes or objects in the software modelling. Constraint is about non-functional properties of systems and related to QoS which captured the fault tolerance, performance, scheduling and task. Real-time behaviour has four categories which are passive, active, event and implementation independence class [15]. A passive class that does not has its own thread of control. An active is a class that has its own thread. An Event class is an active class where behaviour is triggered by event. Implementation independent class is a class whose real-time behaviour can only specified during the implementation phase. Both POAD and ROPES have real time behaviour. In POAD, real time behaviour is placed in class classifier while in ROPES, real time behaviour is placed under Task elements. This is because ROPES has Resource elements and manages all task and real time behaviour of the system. Table 3, Table 4 and Table 5 shows the mapping metamodel of POAD and ROPES.
Table 3. The Similarity of POAD and ROPES Metamodel

<table>
<thead>
<tr>
<th>UML Metamodel</th>
<th>POAD</th>
<th>ROPES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relationship</td>
<td>Relationship defines the association or dependency between classes or objects in the software modelling</td>
<td>Same</td>
</tr>
<tr>
<td>Constraint</td>
<td>Constraint is about non-functional properties of systems and related to QoS which captured the fault tolerance, performance, scheduling and task</td>
<td>Same</td>
</tr>
<tr>
<td>Real-time Behaviour</td>
<td>Consists active, passive, event and implementation independent class</td>
<td>Same</td>
</tr>
<tr>
<td>Pattern</td>
<td>a set of collaborating classes that solves a specific design problem</td>
<td>Same</td>
</tr>
<tr>
<td>Class</td>
<td>A specification for a set of objects and exist at design time</td>
<td>Same</td>
</tr>
</tbody>
</table>

Table 3 illustrates the metamodel mapping between POAD and ROPES methodology. From the table, there are five UML metamodels which are having same semantic description for both methodologies. The five UML metamodels are Relationship, Constraint, Real-time Behaviour, Pattern and Class. These elements are used in constructing a software model in POAD and ROPES methodology. However, there are two UML metamodel elements that have different definition between two models which are Package and Component as shown in Table 4.

Package is general purposes UML modelling for grouping the system elements. They provide a hierarchical view and visibility mechanism to control the visibility of the elements of the package. In POAD, packages are used in modelling patterns. Pattern is a collection of collaborating class that solves a specific problem. Using packages to model patterns requires additional stereotyping and different definition of relationship and interfaces. However, in ROPES, packages are used to represent the domain analysis of the system and each package is usually a part of another domain.

Component represents a modular, deployable and replaceable part of a system that encapsulates implementation and exposes a set of interfaces [16]. In ROPES, a component is used to represents physical packages that used to model source code and physical database. Thus, physical packaging is used to define implementation deliverables. Unlike ROPES, in POAD patterns

Table 4. The Differences of POAD and ROPES Metamodel

<table>
<thead>
<tr>
<th>UML Metamodel</th>
<th>POAD</th>
<th>ROPES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>Pattern is grouped as Package but requires additional stereotyping to reflect the pattern type and relationship between patterns.</td>
<td>Package is a collection of ModelElements and can be represented in domain analysis or subsystem.</td>
</tr>
<tr>
<td>Subsystem</td>
<td>NA</td>
<td>subsystem is generalization of Package and represent the subsystem of the system.</td>
</tr>
<tr>
<td>Component</td>
<td>POAD defines component as design components which is composed a set of classes at the design phase.</td>
<td>Component in ROPES is physical packages used to model source code or physical database.</td>
</tr>
</tbody>
</table>

Table 5. Mapping POAD to ROPES Metamodel

<table>
<thead>
<tr>
<th>UML Metamodel</th>
<th>POAD</th>
<th>ROPES</th>
</tr>
</thead>
<tbody>
<tr>
<td>PatternInterface</td>
<td>A set of classes and operations that define how the pattern interfaces with the other patterns</td>
<td>NA</td>
</tr>
<tr>
<td>InterfaceBinder</td>
<td>It used to connect PatternInterface with its internal classes</td>
<td>NA</td>
</tr>
<tr>
<td>PatternAssociation</td>
<td>It shows the relationship between pattern interfaces</td>
<td>NA</td>
</tr>
<tr>
<td>PatternDependency</td>
<td>It shows the relationship between patterns</td>
<td>NA</td>
</tr>
</tbody>
</table>

Component represents a modular, deployable and replaceable part of a system that encapsulates implementation and exposes a set of interfaces [16]. In ROPES, a component is used to represents physical packages that used to model source code and physical database. Thus, physical packaging is used to define implementation deliverables. Unlike ROPES, in POAD patterns
are visualized as logical packaging and used to organize a set of collaborating classes.

From the metamodel mapping process, the integrated metamodel of POAD and ROPES can be constructed by plug in POAD metamodel into ROPES metamodel at the respective level.

In this paper, we use ROPES as main methodology. The integrated metamodel shows that POAD UML metamodel was added into ROPES metamodel. Pattern metamodel in ROPES is replaced by POAD metamodel. Then, Class metamodel is transformed under the Pattern classes to show that every class construction will be illustrated as pattern. Then, the real time behaviour in POAD is integrated with ROPES.

The main issue in the integrated POAD and ROPES metamodel is the definition of Package and Component. After the integration of both metamodel, the definition of Package and Component in POAD and ROPES still not change. Packages are used to illustrate pattern and domain analysis but they have different semantics depending on it uses. Furthermore, pattern cannot be modelled using UML component because they are conceptually different encapsulation. Hence, it is semantically incorrect to model pattern using component diagram. Because of this restriction, an enhancement of ROPES methodology will use component as logical (pattern) and physical views.

4. DISCUSSION
This paper presents the integrated POAD and ROPES metamodel that aim to enable pattern-oriented development that subject to ERT system. ERT system is the crucial software due to its properties. In addition, ERT system highly dependent on time constraint, task distribution and concurrency resource sharing [17]. Therefore, ERT system development process is difficult because of the complexity of real time properties. This result was motivating the uses of software reuse and for the consequences; patterns will play the main roles to enable the reusability and increasing the software quality as well as decrease the software development cost and time.

There are two important elements in this paper that need to be considered. The elements are 1) POAD and ROPES metamodel and 2) POAD and ROPES metamodel integration. The UML metamodel is used as a modelling languages to defines the metamodels for both methodology. POAD enables the semantic description in their metamodel by defining the Pattern name, PatternInterface, PatternDependency, Class, Constraint, Relationship, InterfaceBinder, PatternAssociation and PatternControlInterface. All of these elements will be used to construct POAD metamodel. The POAD metamodel have been enhanced [15] from the original metamodel in order to capture real time behaviour requirements.

ROPES define the semantic description in their metamodel by using UML metamodel for structural, behaviour, real-time properties and Core profiles.

The reasons of creating metamodel for both methodologies are to identify the semantic description of both models. By doing this, the integrated metamodel of POAD and ROPES can be derived. However, before both metamodel can be integrated, the mapping metamodel for both POAD and ROPES need to be considered. In addition, the mapping the metamodel can ensure what or which level is needed to be integrated as shown in Table 3.
The effectiveness of this integrated metamodel are to motivate the use of pattern early at analysis phase and systematically derived until the design phase that subject to ERT system. ROPES are a methodology that covers both structural and behavioural elements as well as real-time properties. The use of pattern at analysis and design development phase can ensure the high software reuse, software quality and decreasing of software development time and cost.

5. CONCLUSION AND FUTURE PLAN
This paper has developed the integrated POAD and ROPES metamodel in order to enable pattern-oriented development for ERT system. The integration takes into account process-related and metamodel-related criteria. This paper also assessed the capability of both methodologies in supporting pattern-oriented concept in the development phases. The result shows the higher software reusability by propose the integrated metamodel of POAD and ROPES. This works enable to enhancing the ERT system development to include patterns in its development process.

An enhancement of development process for pattern-oriented software development can be developed base on the integrated POAD and ROPES metamodel. From this evaluation, the integrated metamodel needs to change the original ROPES development process in order to enable systematic development process using patterns. Our future research’s focus is to define a software process for integrated metamodel and evaluate how current UML for real-time tools can support the new integrated methodology.

6. ACKNOWLEDGMENTS
The authors would like to thank Software Engineering Department staff, FSKSM for their support.

7. REFERENCES


