Selecting and Formalizing an Architectural Style: A Comparative Study

Ashish Kumar Dwivedi1, Santanu Kumar Rath2
Department of Computer Science and Engineering
National Institute of Technology
Rourkela, 769008, Odisha, India
Email: shil2007@gmail.com1, skrath@nitrkl.ac.in2

Abstract—An architectural style is characterized by their control-flow and data-flow patterns, appropriation of functionality across components, connectors, ports, and roles. Selecting an architectural style for an application is not an easy task. It is a decision-making task in which different objectives must be taken into consideration. In this study, different architectural styles are classified on the basis of many style-induced architectural properties. This classification provides additional guidance for people who lack expertise and experience to select an appropriate style for their software systems. Subsequently, an appropriate style is selected for the case study i.e., cargo router system. After selecting an appropriate style for cargo router system, it is formalized using formal modeling languages Alloy and Promela. For the model checking of these formal notations, automated verifiers such as Alloy Analyzer and SPIN are used. At the end of this study, comparison of performance between modeling languages Alloy and Promela as well as associated tools such as Alloy Analyzer and SPIN is performed.

Index Terms—Architectural Style, Formal Methods, Alloy, Alloy Analyzer, Promela, SPIN, Cargo Router.

I. INTRODUCTION

Software architecture is a set of principal design decisions that deals with high-level structure of a system [1]. In architectural development process, design decisions are represented in terms of structure, behavior, interaction, and non-functional properties of the system. An architectural style is a set of architectural design decisions to capture knowledge of effective designs for achieving specified goals in a given development context [2]. Styles provide a common semantics for a software architect and helps in understanding an architectural design. An architectural style is useful from both prescriptive and descriptive point of view. It simplifies the constraints for different architectural elements i.e., components, connectors, and the formal relationships among those elements. Many styles of software architecture, such as client-server, virtual machine, pipe-and-filter, event-based, rule-based, blackboard, publish-subscriber, peer-to-peer etc. are being applied for development of different application systems. As the complexity of system increases, large number of complex styles are being considered such as C2 (components and connectors), CORBA (Common Object Request Broker Architecture), REST architecture etc.

To endorse architecture based development, formal modeling notations and model checking tools are found to be useful. A number of architectural description languages (ADLs) are available such as Aesop, C2SADEL, ArTek, Darwin, SADL, Rapide, Wright, UniCon, Weaves etc. for modeling and development of software architecture [3]. These ADLs support mathematical notations and tools for modeling different architectural styles and architectural patterns. For example, Rapide [4] is used to model component interface and external behavior of a system; whereas, Wright [5] models an architectural element connector. The tools supported by the ADLs have certain limitations in terms of modeling, visualization, platform support and formal verification. The ADLs are not sufficient for modeling and visualization of complex architectural styles. These complex architectural styles provide semi-formal notation for modeling of complex systems. Hence formal methods are being considered for modeling, refinement, and formal verification of software architecture.

Formal methods intend to describe the software requirements unambiguously and rigorously using different modeling languages and tools that can capture the essential features of a system [6]. In the process of formal modeling, analysis confirms to the consistency of the requested configuration with respect to a particular style. In order to check the compatibility among components, connectors, and configuration, it needs to verify and validate the model. A number of analysis techniques are available for testing, model checking, and evaluating non-functional properties based on the architectural styles. By model checking, important system properties like functional behavior, consistency of internal structure, timing behavior, and performance characteristics are verified.

The goal of this study is to select an appropriate architectural style for a case study i.e., cargo router system [1]. After designing of cargo router system using a particular style, analysis is performed using formal modeling languages Alloy [7] and Promela [8]. Subsequently, Alloy and Promela notations are verified using model checkers Alloy Analyzer [9] [10] and SPIN [11]. Alloy is able to specify structural and behavioral aspects of an architectural style. Alloy is helpful to model desired specification properties such as invariants, inheritance, composition etc. that is compatible with our needs. Promela is based on programming language C and verifies system’s properties in the form of temporal logic. The cargo router system is a logistic system which dispenses received cargo from a list of delivery ports to different warehouses. These cargo are delivered by different vehicles. A number of architectural styles are available for designing an application.
But there is no single style available that is used for all types of problems. Hence, many styles came into existence for designing a particular problem. If a style is selected for the designing of an application, it needs to be formalized. A number of formal models are available in literature for simple styles such as client-server, publish-subscriber, pipe and filter, event-based etc. But less amount of work has been done on formalization of complex architectural styles such as CORBA (Common Object Request Broker Architecture), C2 (component and connector), and REST (REpresentational State Transfer) architecture.

II. RELATED WORK

Galster et al. [12] presented a method called SYSAS, for the systematic selection of architectural styles. In SYSAS method, style selection is based on two process. First characteristics emphasizes basic architectural elements (components and connectors) and second characteristics of target system highlighted the parameters that are visible to end user. Authors have considered a case study to illustrate SYSAS and its applicability. Morisawa et al. [13] developed described architectural styles and a method to select a suitable architectural style for characteristic charts to visualize their characteristics of architectural styles. Authors verified their approach using a real application system.

Wong et al. [14] presented a technique to support the design and verification of software architectural models using the model checker Alloy Analyzer. They have proposed the use of architecture style library in modeling and verifying a complex system that utilizes multi-style structures. Kim and Garlan [15] have mentioned about mapping of an architectural style into a relational model. They expressed an architectural style using formal modeling language Alloy that can be checked for properties such as whether a style is consistent, whether a style satisfies some logical constraints over the architectural structure, and whether one style refines another or not. But they have proposed formal models for simple architectural styles. Bertolino et al. [16] illustrated software architecture-based analysis, evaluation, and testing. In this editorial paper authors report on those that we consider the most relevant advances in the field of architecture-based testing and analysis over the years.

III. PROPOSED WORK

The research work is carried out in this paper is categories into five sections such as selecting an appropriate style, design of cargo router using C2 style, formalizing C2 style using Alloy, formalizing C2 style using Promela and Comparison of performance between Alloy and Promela.

A. Selecting an appropriate style

To choose a suitable style for a particular application is not an easy task. It is based on the type of the application as well as quality parameters of different architectural styles. Different architectural styles are used for different purposes such as batch-sequential (BS) style is used in transaction processing in financial systems. Pipe-and-filter (PF) style is used in operating system application programming. Similarly, virtual machine (VM) is used in the design of operating system as well as network protocol stacks. Client-Server (CS) style is used where an application require centralization of data also where processing and data storage benefit from a high capacity machine. Publish-Subscriber (PS) style is used for GUI programming, multi player network-based games, and news dissemination. Event-Based (EB) style is used for user interface software as well as wide-area application such as financial markets, logistics, sensor networks etc. Peer-to-Peer (PP) style is used where sources of information and operations are distributed and network is ad hoc. C2 (Component and connector) style is used in reactive, heterogeneous, distributed system. CORBA (Common Object Request Broker Architecture) style is used in the creation of distributed software systems composed of components running on different hosts. This style is also used to integrate software components written in different programming languages.

<table>
<thead>
<tr>
<th>QPs / Styles</th>
<th>BS</th>
<th>PF</th>
<th>VM</th>
<th>CS</th>
<th>PS</th>
<th>EB</th>
<th>PP</th>
<th>C2</th>
<th>CORBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficiency</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>*</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Complexity</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>+</td>
<td>++</td>
<td>-</td>
</tr>
<tr>
<td>Scalability</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Heterogeneity</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>0</td>
<td>++</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Adaptability</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td>0</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Portability</td>
<td>0</td>
<td>++</td>
<td>0</td>
<td>++</td>
<td>+</td>
<td>0</td>
<td>++</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Reliability</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>++</td>
<td>+</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Security</td>
<td>0</td>
<td>0</td>
<td>++</td>
<td>+</td>
<td>0</td>
<td>+</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

In this study, an attempt has been made to classify different architectural styles on the basis of quality parameters such as efficiency, complexity, scalability, heterogeneity, adaptability, portability, reliability, and security. These characterizations are mentioned in table 1. In this table the symbol "++" represents that an architectural style performs very well with some specific quality attribute. The symbol "+" represents that a style performs some support for a particular quality attribute. The symbol "0" indicates that a style does not affect quality attribute. The symbol "+" stands for the style has negative impact on some specific quality attributes. These notations are helpful for better understanding of the categorization and evaluation process for different architectural styles [12]. The supportability of these quality parameters varies from one application to other. But these types of characterizations among different architectural styles help an architect for high level design of a software.

From table 1, it is clear that an architectural style C2 supports large number of quality parameters. In this study,
cargo router system has been considered as a case study for which an appropriate style has been selected. Cargo router system has many components that are organized in a layered structure. C2 style is based on layered style having large number of extra features. C2 style is found to be helpful as it provides, substrate independence, accommodating heterogeneity, support for product lines, ability to design in model-view-controller pattern, support for distributed applications etc. Hence C2 style is considered for cargo router system.

B. Design of cargo router system using C2-style

Figure 1 shows the architecture of cargo router system in C2-style. The components in this style are organized in a layered structure. In this example four components such as clock, port, warehouse, and vehicle are placed at the top layer of the cargo router system. The component clock generates a pulse when any other components change their states. The port component is used for keeping track of the state of the ports. Similarly the vehicle component and warehouse component are used to keep track of the state of the vehicles and warehouses respectively. At the next top layer telemetry component is placed. It determines when cargo arrives at the port, and tracks the cargo from the time it is routed until it reaches to the warehouses.

The principle of C2 style is to provide limited visibility among components. A component is familiar with the other components above it in the hierarchy. A component is totally unfamiliar with components beneath it. A component placed on the bottom layer utilizes the services of the components above it by sending request messages. Components at the upper layer emit the notification messages when they change their states. C2 connectors broadcast notification messages to every component and connector present at the bottom layer. These messages represent an implicit invocation mechanism and component’s state change. The components port artist, warehouse artist, and vehicle artist are used for graphically depicting the state of their respective abstract data types to the users. Cargo router is the main component that handles routing requests initiated by the end user. Finally at the most bottom layer GUI component is placed. This component transfers the notifications sent from the artists components on GUI.

C. Formalizing the behavior of C2-style using Alloy

Specifying a model of software application using Alloy has several advantages. First, presenting these formal models in an executable form ensures that model has unambiguous and testable semantics. Second, Alloy visualizes a model of unbounded size and later specifies a size in a bounded form when verifying properties. Automated tool, Alloy Analyzer translates high-level, declarative, relational expression of the formal model into a SAT instances that can be solved by SAT solvers. To make the explanation more precise formal modeling language Alloy is used for specifying essential properties of the cargo router system represented in C2-style. Behavioral properties of the system can be expressed as a form of logical predicates which can be checked by Alloy Analyzer. A C2-style represents a family of architectures that share a common structural organization. In this Alloy model, CargoRouter represents the whole system in terms of components, connectors, and C2-connectors. Each component has a set of ports to connect with different connectors having set of roles. The field holder in the port and role signatures indicates that each role and each port have single holder; because a port is hold by single component and a role is hold by single connector.
In modeling language Alloy, analysis is a form of constraint solving. Analysis encourages the architect, by giving concrete examples that reinforce intuition and suggest new scenarios. It can disclose subtle flaws that architect might not have discovered some how. By adding fact statements, checking assertions and executing a predicate, the analysis problem can be reduced. A fact is a logical constraint that should always hold. In this model many facts have been specified. In the first fact, a constraint indicates that if a component contains a port means, this port is hold by the component and the component is the holder of this port. Similarly, a connector contains a role means, this role is hold by the connector and the connector is the holder of this role. Second fact indicates that if some roles are related to some ports then these roles should be specified by some connectors.

According to the principle of C2-style, two components can communicates with each other through a C2-connector. Actually, C2-connector is not an ordinary connector; it is a combination other connectors. Hence, it can be observed that C2-connector act as a component having set of ports to connect with simple connector having set of roles. Third fact insure that if any component contains a port then that port should not be owned by any other component.

In Alloy, a predicate describes a set of states and transitions, by using constraints among signatures and their fields. Without using predicate, instances cannot be generated for operation except from counterexample. In this study, two predicates such as roleConnectPort and compConnectC2Conn have been considered to specify port-role connection and component-c2connector attachment operations. First predicate is used for a port and role, returning true if they are directly connected. Second predicate, insure the attachment of component and c2connector. The keyword disj is used to restrict the bindings and include ones in which the bound variables are disjoint from one another. In the second predicate, first predicate is directly used. Because predicates are used as built-in functions and it can be easily used by other predicates and facts.

Alloy Analyzer generates instances by executing predicates. In this Alloy model, predicate compConnectC2Conn is executed by using run command for the scope value three. The value of scope indicates that Alloy Analyzer generates at most three instances of each atom. The number of instances generated by Alloy Analyzer can be changed by modifying scope value in the run command. If the value of scope is not specified, Alloy Analyzer assumes the value of scope as three. From the experiment, it is observed that if scope is greater than or equal to seven, Tool presents all types of relationship between the objects.

D. Formalizing the behavior of C2-style using ProMelA/SPIN

PROcess MEta LAnguage is C-like modeling language that allows dynamic creation of concurrent processes. Large numbers of modeling languages are available which specify system’s properties in terms of assertions. However assertions are not sufficient to specify the correctness properties of a system. Hence, temporal logic is being used to specify the correctness properties of C2 style. Promela supports temporal logic for the specification of system properties. The following code represents Promela specification of Cargo Router System. In this model, a data type channel is used having two operations i.e., request and notify. This data type is associated with a mtype variable. Promela supports up to 255 channels. In this Promela model, two identical processes called as Sender have been considered. These processes can be distinguished using a keyword _pid. In this process, request is received by top level components through replyChannel. Other process is Receiver having four identical process. In Receiver process notification is received by bottom level components. In Promela specification, _ used for sending data through channel and _ used for receiving data through the channel. A process can instantiated using run keyword.
The properties of a cargo router system are specified using linear temporal logic (LTL). Following formula indicates that, if a sender sends a request, it will receive eventually. In the second formula, different variables are considered such as bCompSR, tCompRR, tCompSN, and bCompRN. These variables indicate bottom level Component Send Request, top level Component Receive Request, top level Component Send Notification, bottom level Component Receive Notification respectively. This formula describes the principle of C2 style. Third formula ensures that Component can connect with Connector or C2Connector but cannot connect with Component.

```
active [4] proctype Receiver() {
  byte bLevelComp;
  end :
  do
    request!_pid, notify[_pid - 2];
    reply[_pid - 2] ? bLevelComp, data;
    printf("\%d\%d\%d", data, bLevelComp, _pid - 2);
  od }
```

Verification result shown in figure 3. In this result, second line represents the name of a technique to handle the state space explosion problem. In a similar way other lines represent terms related to verification process, such as never claim, assertion violation, states, transitions, memory used, and elapsed time etc.

SPIN executes Promela code in four modes such as random simulation, interactive simulation, verification, and trail mode. In the process of verification, SPIN searches the entire state space for counter example. SPIN performs verification in three steps. First it generates the verifier written in C using Promela. Second it compiles the verifier using the C compiler. Finally it executes the verifier. SPIN inbuilt a tool called SPINSPIDER for generating state transition diagram of a PROMELA code.

IV. COMPARISON BETWEEN THE METHODS

Large numbers of formal modeling languages are available in literature. In this approach, Promela and Alloy are considered for formal description of an architectural style C2 using cargo router system. Because, Promela and Alloy are executable in nature and useful for test driven development. Although, large number of model checkers are available in literature for specification, simulation, and verification. This study considered Alloy Analyzer and SPIN for simulation and formal verification. Like SPIN, SMV/NuSMV [17] (Symbolic Model Checker) tool is also used for model checking process. SMV/NuSMV uses temporal logic for specifying correctness properties. There are two limitations of this tool. First it verifies synchronous systems, second the analysis of any system using SMV is limited due to the infinite state space explosion problem. SPIN can also be used for asynchronous systems and it reduces state space explosion problem by using different optimization techniques such as partial order reduction method. Alloy and Promela support some model transformation tools. UML2Alloy [18] is a model transformation tool that generates Alloy code from UML class diagram. But promela code can not be straightforwardly produced. There is a tool vUml [19] that uses SPIN to perform the verification process. But these model transformation tools are not ready for industrial application.

Alloy Analyzer and SPIN perform the verification process in terms of application scope, memory used, time elapsed, reachable state space, automated reasoning, state structure, a number of state variables used etc. These two model checkers are used for specification, simulation, and verification. Alloy Analyzer and SPIN generate the output in the form of time elapsed and memory used. For model checking, SPIN automatically
checks LTL formulas. Alloy analyzer can only check safety property without using temporal operators. In Alloy Analyzer, temporal sequences are not built in and not optimized. Hence, Alloy can only explore short traces and model checker SPIN explores all traces. For modeling states, Alloy Analyzer and SPIN Tools are bounded. Temporal sequencing can easily be built in Promela using SPIN, but it is not built in Alloy. Alloy allows large number of operators such as “=” and “)” (predicate operator) for automated reasoning but Promela do not support predicate logic. Soundness property of any specification language ensures that they are derived from the formal analysis of a system which should be preserved in an implementation phase. Alloy and Promela are similar to programming languages. Hence, soundness properties can preserve during implementation phase. For the verification point of view, comparison between Alloy Analyzer and Promela/SPIN are shown in table 2.

### V. Conclusion and Future Scope

This study presented a library of style using formal modeling language Alloy to assist the reuse and extensible modeling of a complex and highly distributed components. Compatibility among components, connectors, and C2-connectors of C2 style has been checked using model checkers Alloy Analyzer and SPIN. In this study, Alloy Analyzer and SPIN chosen because Alloy provides a compact model that allows the verification of structural and behavioral properties of a system and SPIN supports temporal operators. Modeling the structural properties of an architectural style has generally been associated with the component-connector abstractions. Formal modeling utilizes architectural description languages (ADLs) for formal description. The use of formal methods in the area of verification and validation gives a platform for development of software and hardware systems by proving the completeness and correctness of models. Formal method is a cost effective technique that is used to reduce the defect rate from software.

### References


