Predicting Architectural Styles from Component Specifications: Extended Abstract

Sutirtha Bhattacharya
PTD Automation
Intel Corporation
Hillsboro, OR – 97124
sutirtha.bhattacharya@intel.com

Dewayne E. Perry
Empirical Software Engineering Lab (ESEL)
ECE, The University of Texas at Austin
Austin, TX 78712
perry@ece.utexas.edu

Abstract*

Software Product Lines (SPL), Component Based Software Engineering (CBSE) and Commercial Off The Shelf (COTS) components provide a rich supporting base for creating software architectures. Further, they promise significant improvements in the quality of software configurations that can be composed from pre-built components. Software architectural styles provide a way for achieving a desired coherence for such component-based architectures. This is because the different architectural styles enforce different quality attributes for a system. If the architectural style of an emergent system could be predicted in advance, a System Integrator could make necessary changes to ensure that the quality attributes dictated by the system requirements were satisfied before the actual system was deployed and tested. In this paper we propose a model for predicting architectural styles based on use cases that need to be met by a system configuration. Moreover, our technique can be used to determine stylistic conformance and hence indicate the presence or absence of architectural drift

Keywords

Component Based Software Engineering, Architectural Style, System Composition, Reuse

1. Introduction and Scope

Software architecture styles represent a cogent form of codification [1, 2, 3] of critical aspects to which an architecture is expected to conform. They differ from patterns in that patterns are the result of a discovery process, not a constraint process. Of course, patterns may play an important role in the creation and specification of a style: commonly occurring patterns provide a useful basis for codification. Part of the confusion comes from the fact that styles can be viewed both prescriptively (i.e., as a complex constraint that must be satisfied) and descriptively (i.e., as a description of what exists).

In 1997 Shaw and Clements proposed a feature-based classification of architectural styles [3]. They proposed that different architectural styles can be discriminated among each other by analyzing the following feature categories.

- Constituent Parts i.e. the components and connectors
- Control Issues i.e. the flow of control among components
- Data Issues i.e. details on how data is processed
- Control/Data Interaction i.e. the relation between control and data
- Type of Reasoning: Analysis techniques applicable to the style

Since different architectural styles support distinct sets of quality attributes, the benefit of evaluating components for suitability to an architectural style is obvious, as the quality attributes for a system are often dictated by the system requirements. The ability to determine the architectural style for a system configuration will help us predict whether the desired quality attributes will be satisfied by the system prior to actual deployment.

In this research we propose a model for documenting component specifications and demonstrate how we can reason over the specifications to determine the emergent architectural style a-priori. The first step in the process is the feature category analysis to ensure that our specification model captures the relevant information that will be used for Style prediction. This is followed by the application of the style prediction algorithm.

2. Approach

The approach for the proposed research is outlined in this section. We start with the assumption that there exists a component repository in which software components relevant for a particular domain have been specified using our architectural specification model. A System Integrator (human) identifies a deployment use-case (made up of a list of services) that needs to be satisfied using pre-built components. For identifying the configuration of components for satisfying the use case, the System Integrator queries the repository for the available components. The reasoning proposed will be done on the set of components returned by the component repository. The envisioned reasoning capabilities will facilitate i)

* This research is supported in part by NSF CISE grant IIS-0438967. Please note that any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation

determining whether the set of components returned by the repository conform to any specific architectural style, and ii) identifying a set of components that conform to a desired architectural style and hence support the desired set of quality attributes.

Our specification model captures an architecture in terms of the architectural elements. These elements are essentially the components and connectors that enable functional partitioning as well as introduce the notion of object orientation. Our model enforces the separation of the functional specs from the non functional specs. The functional specifications are captured in terms of the Interface Spec (captures the interface information for the services provided by the architectural element), the Attribute Specs (captures the domain data supported) and the Behavioral Specs (captures the state transitions supported). The non-functional specs are captured in terms of the Quality Attribute constraints and the Deployment constraints

Using the specification model, we analyze the various feature categories proposed by Shaw and Clements to ensure the information needed for architectural style reasoning is comprehended. We start with the constituent elements of a configuration. This is followed by the Control Issues, the Data Issues and finally the Control/Data interactions.

Based on the feature category attributes the emergent architectural style is predicted, using the Shaw Clements classification. The prediction is based on the values of the feature category attributes determined during the feature category analysis.

The step-by-step process for predicting the emergent architectural style is outlined below:

Step 1: The System Integrator specifies a use case/scenario for which a software configuration needs to be built

Step 2: For each service in the use case, we identify the best fit candidate from the component repository and build the *Base Component List*.

Step 3: For each component in the *Base Component List*, we make a note of its *Component Type* Attribute. If all the components are not of the same type, we consider the component type of the set of components to be the one that is most common.

Step 4: For each component in the *Base Component List*, we make a note of the Connector Type attribute. If all the connectors are not of the same type, we consider the connector type of the configuration of components to be the one that is most common.

Step 5: We determine the Control Topology of the set of components by developing the Control Flow List

Step 6: We determine the Control Synchronicity of the configuration of the components

Step 7: The Data Topology of the configuration of components is determined by developing the Data Flow List

Step 8: The Data Continuity of the configuration is determined

Step 9: We determine whether the Control and Data Topologies are isomorphic

Step 10: From the feature category attributes derived in Steps 3 to Step 9, we reference the Shaw Clements classification to determine the Architectural Style. If no clear conclusion can be drawn, we try to determine the most probable architectural style by considering the maximum number of feature category attributes that can be used in making a prediction that is consistent with the classification

3. Conclusion

We propose an approach for reasoning about architectural styles using component specification and a use case scenario which the system integrator desires to satisfy by using a configuration of components. Using this approach, the system integrator will be able to evaluate several deployment options and the associated implications to the quality attributes before the system has been built. This could prove to be an invaluable way of assessing the final system behavior a-priori.

Given that we can determine the emerging stylistic characteristics of a configuration (whether global or "regional") and determine how close it comes to satisfying a particular architectural style, we can use our approach to determine the conformance of that configuration to particular style. This will be particularly useful during the evolution of a system to detect either architectural drift, or even architectural erosion [1, 7].

We envision this research to evolve, resulting in tools that would make the System Integrator's job easier and more efficient.

4. References

[1] Perry, D. E., Wolf, A. L., "Foundations for the Study of Software Architectures", ACM Software Engineering Notes, 17, 4, October 1992, 40-52

[2] Abowd, G., Allen, R., Garlan, G., "Using style to understand descriptions of software architecture", Proceedings of the 1st ACM SIGSOFT symposium on Foundations of software engineering, 1993, 9-20

[3] Shaw, M., Clements, P., "A Field Guide to Boxology: Preliminary Classification of Architectural Styles for Software Systems", Proceedings of the 21st International Computer Software and Applications Conference, 1997, 6-13

[4] Habermann, A. N., Perry, D. E., "Well Formed System Composition. Carnegie-Mellon University, Technical Report CMU-CS-80-117. March 1980

[5] Bhattacharya, S. "Specification and Evaluation of Technology Components to Enhance Reuse," Masters Thesis, The University of Texas at Austin, July 2000

[6] Bhattacharya, S., Perry, D. E.,. "Contextual Reusability Metrics for Event-Based Architectures", The 4th International Symposium on Experimental Software Engineering, November 2005, Australia

[7] Perry, D. E., Wolf, A. L., "Software Architecture", August1989. http://www.ece.utexas.edu/~perry/work/papers/swa89.pdf