Some Initial Performance Characteristics of Three Architectural Styles

Håkan Grahn and Jan Bosch
Department of Computer Science, University of Karlskrona/Ronneby
Soft Center, S-372 25 Ronneby, Sweden
[Hakan.Grahn|Jan.Bosch]@ide.hk-r.se, http://www.ide.hk-r.se/[~nesse|~bosch]

1. Introduction

The design of the software architecture [1, 4] of a software system is concerned with the careful balancing of, possibly conflicting, quality attributes such as performance, maintainability, reliability, and flexibility. One design decision is the selection of an appropriate architectural style [2, 4], since each style provides different levels of support for quality attributes. However, understanding of the characteristics of different styles is still largely qualitative and little quantitative data exists. This paper presents an initial evaluation of the performance characteristics of three architectural styles, i.e., the pipes & filters, the layered, and the blackboard style. The evaluation was performed using simulations and includes both analysis for each style as well as a comparative analysis between the styles.

2. Software Architecture and Styles

The design of software systems has traditionally been centred around the functional requirements. Notable exceptions to this are the research communities that focus on a single quality attribute, e.g., the real-time and the reusable software communities. However, the balancing of several quality attributes has been only little addressed by software engineering research.

Software architecture [2, 3, 4] has emerged as an important research area, since the architecture of a software system constrains the quality attributes. Thus, architectures have theoretical and practical limits for quality attributes that may cause the quality requirements not to be fulfilled. Further, the architecture of a software system is fundamental to its structure and cannot be changed without affecting virtually all components, which may require considerable effort for a system already implemented.

An architectural style [2, 4] defines a family of systems in terms of a pattern of organization, i.e., the types of components and connectors, and the constraints. Styles can be organized into five categories [1], independent components, data-flow, data-centred, virtual machine, and call-and-return. Each category contains a number of styles. The styles considered in this paper are pipes & filters, layered, and blackboard, that are members of the data-flow, call-and-return, and data-centred categories, respectively.

In the pipes & filter style, each component has one or more inputs and one or more outputs. Streams of data are read from the inputs, processed, and streams of data are produced on the outputs. Since, generally, the component starts generating output before all input data is received, the components are generally referred to as filters. The connectors are often referred to as pipes.

Layered systems are organized hierarchically with each layer providing services to the higher layer and using services of the layer below it. Ideally, each layer in the system provides a level of abstraction exposing only those aspects relevant on this level of abstraction. Some layered systems, so called non-pure, break the rules and allow layers to call any lower, or even a higher, layer.

The blackboard style defines two types of components, i.e., a central data structure that maintains the public part of the system state and a set of independent components (clients) that operate on the central data structure. The components generally have no or limited interaction between each other, but use the data structure as a means for interaction.

3. Characterisation Methodology

Architectural design is concerned with the balancing of quality requirements. Unfortunately, little knowledge exists about the precise characteristics of each architectural style and how these characteristics vary when variables of the style are changed. To address this, we try to answer the following questions:

- For each studied architectural style, what are the variables of the style that influence the performance and what are the performance characteristics?
- How do the performance characteristics of the studied architectural styles relate to each other?

Our intention is to present relative performance characteristics, that allow the software architect to compare different architectural styles and variations within a style, e.g., changing the number of components or the way they interact.

We first define the parameters (or variables) that influence the performance characteristics of an architecture based on the architectural style. The general, i.e., style independent, parameters that we use in our evaluation are: *Number of components*; *Computation vs. communication ratio*; and *Use of blocking or non-blocking communication*.

In addition to the general parameters, some style specific parameters are identified as well. Several sub-styles to the pipes & filters style exists [1, 4], which differ in the way filters are connected, e.g., one-to-one or one-to-many, and the organisation of the computation, i.e., batch-sequential vs. concurrent. The primary variation of the layered style is whether the architecture is purely layered or not. A second aspect is the number of layer stacks in the system. One of the aspects of the blackboard style is how the control flow is implemented. In general, some component updates the blackboard, which causes the blackboard to notify all affected components. The number of updated components is an important parameter related to performance.

When evaluating the performance, we make use of four performance indicators: *Throughput*, *System response time*, *Queue time for events*, and *Queue length for components*. The throughput and the system response time indicate the overall performance of the system, while the other two metrics provide information

about the internal behaviour of the system, e.g., contention, possible performance bottlenecks, and starvation.

Our approach to characterising the performance of the architectural styles is *simulation* of abstract software architectures based on the each style. An architecture is described as a set of components that send directed events to each other.

Each component in the architecture has a type. The component type specifies the interface and the required acquaintances of its instances. The interface consists of a set of interface elements, called event handlers. Each handler defines the behaviour of the component in response to the receipt of an event. For each arriving event, the corresponding event handler is executed. The 'execution' consists of events sent to other components and a fixed amount of execution time. Events are divided into a *before-set*, i.e., the events that are sent before the operation is 'executed', and an *after-set*, i.e., the events that are sent afterwards.

Since event handlers send events to other components, the component type defines a set of acquaintances that act as placeholders for the actual components. Thus, the events in the before- and after-set refer to acquaintances instead of actual components. Upon component instantiation, the acquaintances are bound to concrete components.

4. Initial Simulation Results

In our simulations, we have execute 100000 events in the system, and then we have interrupted the execution and calculated our performance metrics. New events arrive to the system as fast as possible, i.e., as fast as the initial (or input) component can process them.

We have varied the number of components for each style: between 5 and 100 filters for the pipes & filters style; 10, 20, 30, 40, and 50 layers for the layered style; and 10, 20, 30, 40, and 50 independent (client) components for the blackboard style. In addition, we have simulated computation to communication ratios of 1:1 (C=1), 10:1 (C=10), and 100:1 (C=100). Due to the page limitation we only briefly discuss our results, primarily focusing on the system response time.

We start our evaluation with the pipes & filters style. We find that the normalised system response time increases proportionally to the number of components, i.e., the system response time increases to the square of the number of components. We have also observed that there seems to be a constant factor of two between the normalised system response time for C=1 as compared to for C=10 and C=100, i.e., when C>=10 the normalised system response time seems to be less dependent on C.

As for the layered style, we have seen the following. The normalised system response time is proportional to the number of components, i.e., the system response time grows relative to the square of the number of components. For C=10 and C=100 the curves for the normalised system response time follow each other closely. We speculate that when C>=10, the communication and queue times have low impact on the system response time, and thus the computation time dominates. Since we normalise the response time with respect to the computation time, the curves will be similar for large computation to communication ratios.

Finally, for the blackboard style we reach the following conclusions. The normalised system response time seems to be independent of the number of clients, i.e., the system response time grows linearly to the number of clients. There is also a constant factor that differs depending on the computation to communication ratio. For C=1 the constant seems be about 4, and for C=10 and C=100 the constant seems to be about 2. For large ratios (C>=10) the curves for the normalised average system response time follow each other closely, similarly to the layered style.

By comparing the performance characteristics of the three architectural styles we have found both differences and similarities. For both the pipes & filters and the layered styles, the normalised system response time increases proportionally to the number of components, i.e., the system response time grows to the square of the number of components. By contrast, for the blackboard style the system response time seems to only grow linearly to the number of components.

We have observed several performance similarities between the different architectural styles. First, the normalised queue time approaches an asymptotic value when the number of components is larger than 20-30, i.e., the average queue time grows linearly to the number of components in the system. Second, there seems to be a constant factor of two in the normalised system response time between C=1, and C=10 and C=100, independent of the number of components in the system. Further, when C>=10, the normalised system response time seems to be almost independent of the exact computation to communication ratio.

5. Concluding Remarks

In this paper we try to characterise the performance of three architectural styles. In order to get performance data we have taken an event-driven simulation approach. The architecture is described in terms of components and directed interconnections where events are sent.

There are several aspects that we have been forced to exclude from this paper, e.g., the impact of blocking communication and non-pure layered styles. These aspects will be reported in another paper. We also currently working on implementation of real applications using different architectural styles and plan to compare their performance to the predictions provided by our simulations.

This early prototype of our simulation toolkit has some limitations. For example, we can only predict the performance of architectures executing on a uni-processor, i.e., parallel and distributed systems are not modelled. Further, we assume infinite communication bandwidth and a fixed latency.

In our future work we plan to incorporate the possibility to evaluate other quality attributes as well, e.g., maintainability and flexibility. The long term goal is to have a tool that can evaluate several quality attributes, especially contradicting ones such as performance and flexibility.

References

[1] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, Addison-Wesley, 1998.

[2] F. Buschmann, C. Jäkel, R. Meunier, H. Rohnert, and M.Stahl, *Pattern-Oriented Software Architecture - A System of Patterns*, John Wiley & Sons, 1996.

[3] D.E. Perry and A.L.Wolf, 'Foundations for the Study of Software Architecture,' *Software Engineering Notes*, 17(4):40-52, Oct. 1992.

[4] M. Shaw and D. Garlan, *Software Architecture - Perspectives on an Emerging Discipline*, Prentice Hall, 1996.