An Object-Oriented Bridge among Architectural Styles, Aspects and Frameworks

J. Andrés Díaz Pace

ISISTAN Research Institute, Faculty of Sciences, UNICEN University
Campus Universitario, Paraje Arroyo Seco, (B7001BBO) Tandil, Buenos Aires, Argentina
Also CONICET - Argentina
Email: adiaz@exa.unicen.edu.ar

It is generally agreed that designing high-quality frameworks results a difficult task, mainly because this process often relies more on the designer's expertise than the technology used to implement such designs [4]. Besides, frameworks should also take into account several quality attributes. Therefore, framework development practices typically involve considerable efforts. This would indicate that some aspects of the domain could not be directly modeled in terms of object-oriented concepts. The problem is how to break the tradeoffs imposed by a pure functional decomposition versus a pure object-oriented decomposition of a system.

In this context, a design approach driven primarily by architectural models focuses on the solution of design problems taking as main driver the organization of software components as a function of the quality attributes affecting the system. In this way, developers should be able to construct a given system by assembling and elaborating certain architectural fragments, as independently as possible of particular implementation technologies. Nonetheless, improved software quality cannot be achieved simply by focusing on isolated architectural styles and their associated quality attributes. In fact, as software complexity gets bigger, designers need to deal with a variety of special computing concerns or aspects [6] (e.g., synchronization, error-checking strategies, resource sharing or usage, distribution and performance, among others) in order to fulfill these quality factors. Particular styles may be able to address specific aspects, but the final system architecture needs to be built considering a number of relevant concerns, which usually present uneven organizations. Moreover, the intrinsic nature of concerns makes it difficult to cleanly segregate them [5]. Once the system architecture is given, the underlying organization of its concerns comes attached. The discussion is not just about new mechanisms or artifacts to handle software concerns, rather it involves how technologies can contribute to effectively reason about concerns by promoting good design practices.

We propose an architecture-driven design approach based on the concept of proto-frameworks [1], aiming to provide an intermediate stage in the transition from architectural models to object-oriented frameworks or applications. The approach is based on an object-oriented materialization of domain-specific architectures derived from domain models, that is the production of concrete computational representations of abstract architectural descriptions using object-oriented technology. A proto-framework materializes, in object-oriented terms, the infrastructure required for cooperation and communication of each architectural component type. In other words, a proto-framework provides the essential abstractions to derive new applications or frameworks by inheritance from the proto-framework classes. In this case, the framework provides very abstract hooks to map specific domain components into a class hierarchy in a white-box fashion. This mapping can produce a specific application, but more important yet, it can produce new domain-specific frameworks that adopt the underlying architectural model. Using an architecture-oriented approach, developers are able to better identify relevant concerns and reason about them at the very conception of the system architecture [3].

The main contribution of the approach is that proto-frameworks make explicit some essential architectural choices by means of object-oriented constructs, which can serve later as basis for the development of either traditional frameworks or final applications. Therefore, the tradeoffs among quality attributes selected by framework developers for this design can determine different alternatives of building frameworks on top of proto-frameworks.

Object-oriented Materialization of Software Architectures

By materialization we mean the process of producing a concrete computational representation from an abstract description using a given technology. Starting with an abstract domain model and one or more architectural styles (e.g. pipeline, blackboard, hierarchical layers, etc.) suitable for the domain, we obtain a first architectural materialization of the domain. This materialization is driven primarily by quality attributes (non-functional requirements) that predominate in that domain. In the proposed approach (see Figure 1), we can basically identify two stages. First, developers should figure out the problem architecture, that is an architecture representing the primary organization of the software to build and the tradeoffs imposed by non-functional requirements and architectural styles. Here, concerns are initially mapped to architectural constructs, instead of be coded using framework or languages constructs. Second, as a result of this general description of relevant concerns, the approach enables a materialization of these concerns into a proto-framework, and then several kinds of framework implementations, whereas these frameworks retain the properties inherited from the original system architecture.

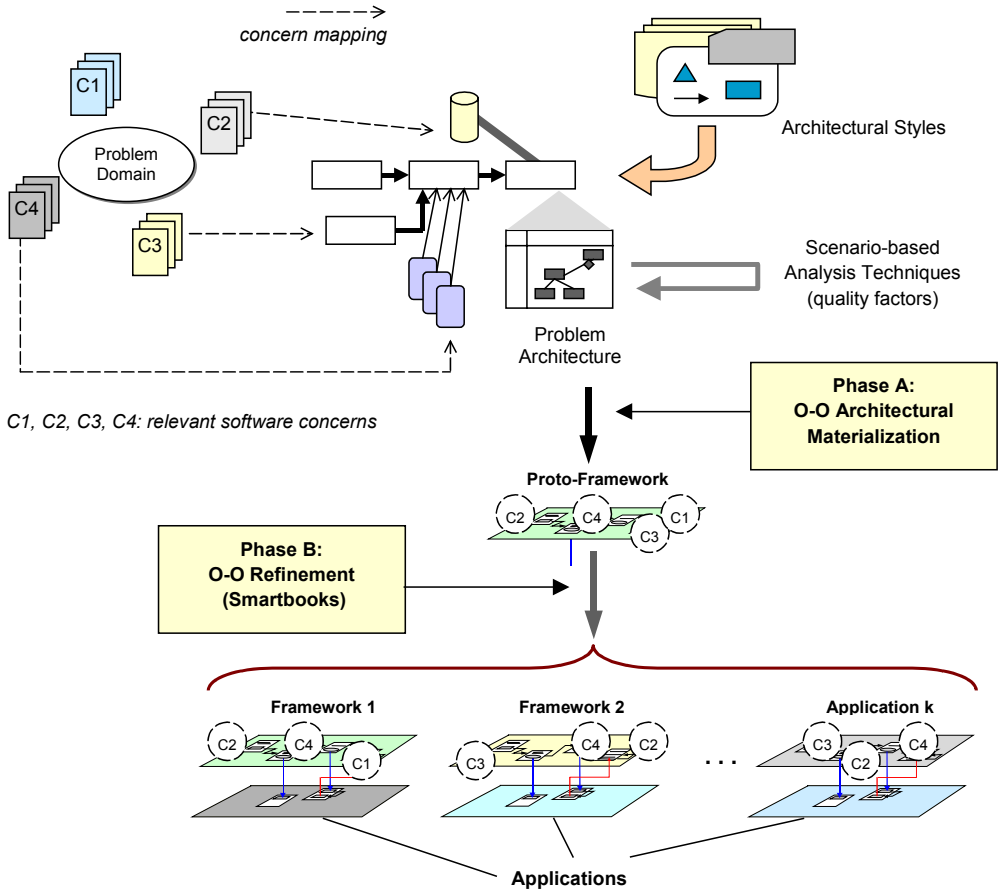

Figure 1. Materialization approach to framework design

In order to make this approach effective, it is necessary to provide techniques assisting developers to bridge the gap between architectural specifications and concrete object-oriented implementations. In this line, we are working on a novel approach called Smartbooks [7], initially conceived to improve framework usability by means of knowledge-driven documentation. This method can be easily adapted to the second phase of our approach because a proto-framework is an object-oriented framework itself. Moreover, we believe this can transcend the framework scope to be applied at higher levels of abstraction. If we insert proto-frameworks, it would be reasonable to count with means to transform architectural designs into proto-framework structures. We are currently investigating the use of smart techniques for aspect-oriented software development [2], so that high-level aspect specifications can be mapped into aspect implementations (for example, aspect-oriented frameworks or aspect languages) by using adequate documentation knowledge. We believe this approach (with some modifications) can be extended to fulfill the requirements of our problem regarding object-oriented materialization of software architectures.

At this moment, we have developed a proto-framework called Bubble derived from previous research in the subject. The model can be informally summarized as a set of cooperating agents, that we call bubbles, which perform certain computations and use an event-based implicit-invocation style to notify other bubbles about their computations. Bubbles are also equipped with associated sensors, which are registered to process certain kinds of events according with predefined relevance criteria. The behavior of any bubble in the system is defined through tasks using a condition-action format. In this way, complex interactions, structures and behaviors can be modeled combining these primary blocks. The current version of the Bubble model has been implemented in Java, and it allows developers to define and configure different kinds of components, such as: entities, groups, tasks, sensors, and interaction protocols. We have partially validated the feasibility of the approach with several applications/frameworks implemented on top of Bubble, namely: a multi-agent simulation of bubbly flow, a framework for Enterprise Quality Management systems (EQM) called InQuality [1], and a soccer simulation. These examples served to compare alternative framework implementations derived from a target proto-framework.

Presentation

The poster presentation will be based on the following topics:

- Frameworks and proto-frameworks
- The role of aspects in software design
- Object-oriented materialization of software architectures
- An example with the Bubble model
- Application of AI techniques to support the approach

References

[1] Marcelo Campo, Andrés DiazPace, and Mario Zito. *Developing Object-Oriented Enterprise Quality Frameworks using Proto-Frameworks*. Software Practice and Experience 32(8): 837-843. Theme Issue on Enterprise Frameworks, 2002.

[2] Andres Diaz Pace, Marcelo Campo and Federico Trilnik. *Assisting the Development of Aspect-based Multi-Agent Systems using the Smartweaver Approach*. Proceedings SELMAS 2002. LNCS Springer Special Volume on Software Engineering for Large-Scale Multi-Agent Systems, 2003

[3] Marcelo Campo and Andres Diaz Pace. *Analyzing the Role of Aspects in Software Design*. CACM 44(10): 66-73 - Special Issue on Aspect-Oriented Programming , 2001.

[4] Mohamed Fayad, Douglas Schmidt, and Ralph Johnson. *Building Application Frameworks, Object-Oriented Foundations of Framework Design*. Wiley Computing Publishing, 1999.

[5] Walter Hürsch and Cristina Videira Lopes. *Separation of concerns*. Technical Report NU-CCS-95-03, College of Computer Science, Northeastern University, Boston, Massachusetts, February24 1995.

[6] Gregor Kiczales, John Lamping, Anurag Menhdhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin. *Aspect-oriented programming*. In Mehmet Ak›sit and Satoshi Matsuoka, editors, ECOOP '97 --- Object-Oriented Programming 11th European Conference, Jyväskylä, Finland, volume 1241 of Lecture Notes in Computer Science, pages 220--242. Springer-Verlag, New York, NY, June 1997.

[7] Alvaro Ortigosa, Marcelo Campo, and Roberto Moriyon. *Towards agent-oriented assistance for framework instantiation*. In Proceedings of the Conference on Object-Oriented Programming, Systems, Languages and Application (OOPSLA-00), volume 35.10 of ACM Sigplan Notices, pages 253--263, N. Y., October 15--19 2000. ACM Press.