A New Approach to Introduce Aspects in Software Architecture

Hadjer Khider
Saad Dahlab University
09000 Blida, Algeria
Khetude@gmail.com,

Djamal Bennouar
Saad Dahlab University
09000 Blida, Algeria
dbennouar@gmail.com

ABSTRACT:

The techniques of programming and methodologies strongly evolved throughout the history of data processing with the evolution of the software systems, these systems indeed tend to become increasingly complex. Component-Based Software Development proved its interests in the control of the complexity of the conceived software, and became a critical factor in the success of development of the software projects by facilitating the maintenance and the evolution of the software and authorizing the development of the bulky systems in terms of size but also of complexity.

This style of programming promises the re-use, but is confronted with the problems of *code scattering and tangling.* The application of Aspect-Oriented Programming on the software components makes it possible to face these problems. Programming called by aspect allowing managing, in a modular way, these concerns by separating them from the basic code.

Aspect-Oriented Programming, a new paradigm of the programming which made possible to simplify the writing of the programs data-processing, while making them more modular and easier has to make evolve.

Today, the software Aspects and components are two very promising paradigms; who support the re-use and simplify the software development. To date, implementation the simultaneous of these two paradigms remains a field of research very slightly explored. To date no model of component supports in an explicit way the aspects and several questions remain open. Among them: How to integrate the representation of the aspects in the software components? How to manage the interactions and overlappings between aspects?

We present in this paper 3ADL, an extension of the model of component IASA defined in the laboratory LRDSI which supports the Aspect-Oriented Programming. This extension consists in equipping approach IASA with the aspect components and aspect ports.

The objective of work is to make supports to the model of component IASA the concept of aspect in its entire dimension: Once this concept supported, an architect could define his own Aspect components which it instantiated in the part controls of a component.

ICPE'11, March 14–16, 2011, Karlsruhe, Germany.

Categories and Subject Descriptors

D.2.11 Software, SOFTWARE ENGINEERING, Software Architectures: Domain-specific architectures

General Terms: Design

Keywords: AOSD, Advice, Aspect, Aspect-Oriented Programming, Component, JoinPoint, PointCut, Software architecture, Weaving

1. INTRODUCTION:

IASA approach is a specific component model for the specification of software architecture. The separation into components and provide a first level hierarchy to modularize a software architecture and provide an affordable according to a variable granularity for the architect. It is then possible, through the hierarchy of the model to imagine an incremental construction starting from the specification of composite greater granularity to the specification of primitive components. Indeed, some concerns can not be properly modularized using such an approach and then find themselves embedded within a software architecture description. That is why we propose in this paper 3ADL extension of component model IASA for the integration of new concerns in software architecture to designing software architecture step by step.

1. THE FUNDAMENTAL CONCEPTS AND MODELS IN IASA

The IASA approach is based on a unified component model oriented to support system design where some components may be deployed as software components and others as hardware components. IASA is based on the following concepts: *access point*, *port*, *component*, *envelope, connector* and *action.*

The component is a fundamental element in defining software architecture. The component model distinguishes between two broad categories of components The *primitive* components and *composite* components.

The component model defines two views in a component, an external view which must adhere to any component and an internal view, applicable only to composites. The external view is represented by the concept of the *envelope* or the ports are located modeling component. The internal view is organized into two main parts: the *operative part* and *control part.*

In IASA, a component interacts with the external world through a set of ports. A port has a structure made of access points and a behavior. A port is the unique place where *joinpoints* are localized. Aspect management is achieved through two main directives*: injection* and *removal*. The aspect injection weaves the

advice type into the behavior of the port and places a connector between the port providing an advice and the advised port.
The instantiation of a component is realized in the context of the envelope concept. An envelope is used to isolate the pure instance of a component of its operating environment by providing it the necessary elements for the operation of the proceeding. In 3ADL, an envelope is used to instantiate a component type. The external view of any component is formed of an envelope having one or more ports.

1.1. The Access Point Concept

The main purpose of the 3ADL *access point* concept is to provide a unified way to represent component's interaction points in the specification of a system architecture using software component and/or hardware components. An access point is either a Data Oriented Access Point (DOAP) or an Action Oriented Access Point (ACTOAP). An ACTOAP represents a service which may support many distinct actions. A DOAP is provided with an attribute specifying the data direction (*in, out,* and *inout*).

2. Aspect Oriented Software Architecture with 3ADL

Any behavioral element of a port is considered as a potential *joinpoint.* The behavioral elements include the actions attached to an ACTOAP, the implicit actions associated with a DOAP (i.e. *send, receive, updated, changed*) and the rules defining the port's behavior. An ACTOAP (*server* or *client*) associated with *Aspect Oriented Actions* is called an ASPOAP (Aspect Oriented Access Point). An ASPOAP server is also called an *advice* ASPOAP. *Aspect oriented actions* are actually represented as a set of predefined aliases called *aspect aliases.* Each *aspect alias* is associated with a specific advice type. The number of actions associated with an advice ASPOAP does not depend on the number of supported advice types. The widely accepted advice type (after, before and around) are supported through five basic aspect aliases: *aroundFirstAction*, *AroundLastAction*, *proceeedAction, beforeAction* and *afterAction.* Once instantiated and connected to an advice ASPOAP, the client ASPOAP is provided with extended aspect aliases. Each extended aspect alias is built by prefixing a basic aspect alias with the name of the port containing the advice ASPOAP and an aspect ordering number (e.g *pAuthAdvice_1_ aroundFirstAction*). This later is used to determine the launching order of advices when two or more aspects are attached to the same joinpoint.

2.2 Advice insertion mechanism

The advice injection modifies the structure and the behavior of advised *ClientPort,* advised *DataPort* and all *ClientPort* connected to the advised *ServerPort*. The advice injection specifies the *pointcut*, an advice type (before, after, around) and a possible mapping of a DOAP of an advice port to a DOAP associated with a *jointpoint*. Usually we use the term *aspectual connector* to refer to the connector linking a Client ASPOAP to an advice ASPOAP. In the 3ADL graphic notations, a dotted line is used to represent an aspectual connector.

```
/// IASA 3ADL: File X25CM.3adl
/// X25CM component type
……
   controlpart {
      components { X25CMOPCtrl starter;}
      connetors {// controlpart's connector s description }
    }
   optionpart { // the keyword aspectpart may be used here
     components { LogCmp logCmp;}
     connetors { //option's connector s description }
       // Aspect management specification
      pointcuts {
       log_alarm = {alarm.receive};
       log_enable = {pEnable.*.receive};
logall = {serverport, clientport, outdataport};
       logfire = {serverport.*.fire};
       // Advise services only, not data
       logpOnServices = logpall – {send, receive};
            // a trace pointcut
             secFTPTrace={FTPClientPort.getTicketFile.rule};
       }
       advices {
         inject LogCmp.log after log_alaena;
         // The previous construct is equivalent to the following
         // inject logCmp.log after log_alaena
         // bind {log} to {alarm}, {enable};
         inject LogCmp.log around logFTPTrace;
       log_alaena = log_alarm + log_enable;
```

Figure 1. Pointcut definition and advice injection

2 Conclusion:

The main difference between AOSA in IASA and in the just introduced approaches is mainly due to the level of abstraction of the basic model elements of IASA. The IASA aspect orientation is achieved at a level of abstraction totally independent from any software mechanism, even from interface concept which represents the fundamental interaction point in the other approaches. In these later approaches, the joinpoints are reduced to the main operations specified at the interface level. Elements involved in such operations cannot be reached and considered as joinpoint. In IASA, all elements present at port level may be considered as potential joinpoints. Hence, implicit actions on DOAP (i.e. *send, receive, updated, changed*) may be considered as joinpoints even if the DOAP is a resource associated with an ACTOAP or is a part of a complex DOAP.

The IASA aspect orientation may be seen as a symmetric or asymmetric AOSA approach. The use of the same component model for aspect space and business space make IASA a symmetric approach. The predefinition of a set of aspect components provided at the port level with a catalogued aspect oriented action may make IASA an asymmetric approach. However, the IASA asymmetry is not against the reusability of an aspect component in the business space, despite the fact that the IASA elaboration process does not recommend such practice. IASA natively supports Aspect Oriented Software Architecture Specification. The concept of aspect is part of IASA main model elements. The concept of separating cross cutting concern represents the key element in the definition of the internal structure organization of the IASA component model.