社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

Webアプリケーション・フィーチャのアスペクト隠蔽

アハロム レダ† 外村 慶二†† ダニエル バルーク† 中島 震† 鵜林 尚靖††

† 国立情報学研究所 〒101–8430 東京都千代田区一ツ橋 2–1–2
†† 九州工業大学 〒820–8502 飯塚市川津 680–4
E-mail: ††keiji@famteam.org, †††nkjm@nii.ac.jp, ††††ubayashi@acm.org

あらまし Webアプリケーションは一般の利用者にとってわかりやすいことから機能変更の要求が多い。プロトタイピング中心の繰り返し開発の方法が採用され、その結果、プログラム構造が悪くなり保守性が低下する。アスペクト指向プログラミングを用いると、横断的な関心事を含む多様な観点からプログラムを良構造化することができ、繰り返し開発と保守性を両立させることを期待できる。本稿ではWebアプリケーション構築にアスペクト概念を適用することで、開発スタイルがどのように変わるかを考察する。

キーワード アスペクト指向プログラミング，Webアプリケーション開発，情報隠蔽

Aspectual Encapsulation of Web Application Features

Reda AHROUM†, Keiji HOKAMURA††, Daniel BALOUEK†, Shin NAKAJIMA†, and Naoyasu UBAYASHI††

† National Institute of Informatics 2–1–2 Hitotsubashi, Chiyoda-ku, Tokyo, 101–8430 Japan
†† Kyusyu Institute of Technology 680–4 Kawatsu, Iizuka-shi, Fukuoka, 820–8502 Japan
E-mail: ††keiji@famteam.org, †††nkjm@nii.ac.jp, ††††ubayashi@acm.org

Abstract Web application, sometimes developed in an incremental and iterative manner, may result in a poor organization of program codes. For a better modularization, the notion of aspect can be introduced. This paper demonstrates the idea with an example case of Web application development by using AOWP, a new AOP framework for PHP programs.

Key words Aspect-Oriented Programming, Web Application Development, Information Encapsulation

1. Introduction

Web application is increasing to provide new style of software systems making use of broad-band networks. They include EC (electronic commerce), SNS (social network service), and others emerging in days to come. Due to new commercial demands and feedback from the users as well, Web applications, or Web-based systems, usually evolve rapidly and are subjected to frequent modifications [2] [15]. For such reasons, an iterative development style based on rapid prototyping and continuous changes are adapted. It may have a risk of leading to a less-organized program structure not amenable to a long-term maintenance. For example, adding a new application feature may result in changes in various Web pages. Although the change is logically small, it actually affects the descriptions of many Web pages. To deal with such scattered changes, the notion of AOP (aspect-oriented programming) is helpful.

AOP, as demonstrated its practicality by AspectJ [5] [11], is now adapted in various programming languages to include PHP. It is a language widely used in Web-based systems, especially for the server-side application program development. Several tools [3] [7] [17] have been so far developed to enable AOP for PHP. AOWP [8] [9], among others, provides Web-specific pointcuts, and thus makes it easy for adaption in developing Web applications. The development is usually centered around designing Web page transitions, and the application logic written in PHP is desirable to be separated from it. Encapsulating application feature in an aspect would make it possible a new style of the development.

Aspect is a notion to encapsulate cross-cutting concern [5]. It is not always homogeneous but sometime heterogeneous

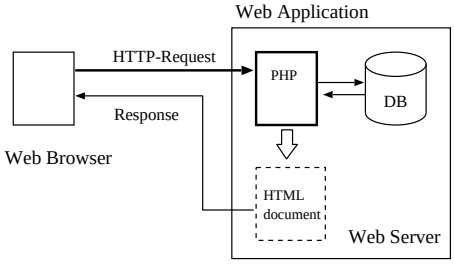

図 1　Web Application Architecture

[4]. Homogeneous aspect, such as the famous *logging*, has a generic pointcut to look at many join points. Same advice codes can be woven into many join points. Heterogeneous aspect, on the other hand, only looks at a small number of join points, just one particular execution point in its extremal case. Although such an extremal one seems no different than just a method or procedure call, it is usable due to the obliviousness [6]. A new application feature may be added as an aspect, while the same aspect is deleted when it becomes obsolete. The aspect is easy to delete since unweaving is as easy as weaving. It does not need any of the impact analysis [16] to have consistent codes.

This paper reports a Web application development practice to make fully use of the aspect-oriented programming techniques. In the development, AOWP, a new AOP framework for PHP, is used where many interesting application features are encapsulated in heterogeneous aspects. Although the discussion would be more like qualitative, the experience shows that the notion of aspect has a great advantage in an incremental style of Web application development.

2. Web Applications and AOP

2.1 Web Applications

Figure 1 illustrates a simplified, but a typical architecture of Web applications. A client uses Web Browser to access a Web Server on which the target application system is running. The system is written in a Web application language PHP, and usually uses back-end database systems to store persistent data. The program, invoked by a client's request, may access the database and generates an HTML document returning to the client. An HTML page is sometimes called a *view* since it provides a graphic interface to the clients using Web Browser.

Web application accepts multiple requests from more than one client at a time, and it also performs *long-term transactions*. Although each HTTP request is independent in view of communication protocol, a series of requests from one particular client is considered to constitute a user-session. Usually, *Cookie* is introduced to keep track of such accesses. The notion of user session is important in Web application since it is the basis for separating multiple users from each other and for having a consistent series of accesses by a particular user.

表 1　Extent of Aspect

Aspect	Description
Per Application	A Single Instance in the Application
Per Session	One for each User Session
Per Request	One for each Request
Per Join Point	One for each join point

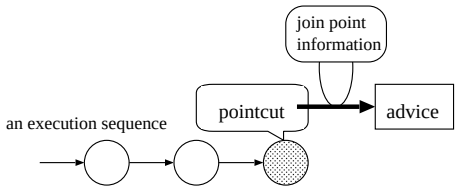

図 2　Pointcut & Advice Model

2.2 AOWP

AOWP [1] [8] is a new aspect-oriented framework for PHP programs. It adapts the join point model that AspectJ has advocated, but provides pointcut designators specific to Web applications written in PHP. Moreover, the aspect instantiation model of AOWP is unique to take into account the characteristics of Web applications. Extent of an aspect may be different depending on its purpose. As mentioned earlier, a Web application accepts multiple concurrent requests and keeps session informations for each client. AOWP provides several extents for aspects as shown in Table 1.

`PerApplication` aspect introduces its single instance for the entire application. Its extent is the same as the application itself. `PerSession` is created for each user session and is suitable for looking at join points of the session. `PerRequest` is active during the execution of a single HTTP request and `PerJoinPoint` is just for a stateless aspect instance.

In addition to the aspect extent, AOWP differs from other AOP languages such as AspectJ [11], GAP [3] or AOPHP [17] in that it provides pointcut designators specific to Web application. They include Web-specific ones such as `HTTP_request`, `cookie_get`, and `cookie_set` as well as those standard ones such as `function_call` of AspectJ.

Figure 2 illustrates the pointcut & advice model, which is basically the same as what AspectJ adapts. When a pointcut designator of an aspect is matched with some point in executions, the accompanying advice code is invoked. The exact ordering of executing the base code and advice is dependent on the choice of the advice strategy such as `before` or `after`. The advice code is invoked with a pre-defined argument of type `AOWP_JoinPoint` to carry the information relating to the location that the pointcut designator matches with. The information is what cannot be easily reconstructed without

```
class UserRightAspect
                extends AOWP_PerSessionAspect
  public function _construct() {
    $verifPC = new AOWP_RequestPointcut(...);
    $verifAdvice = new AOWP_BeforeAdvice();
    $verifAdvice->setPointcut($verifPC);
    $verifAdvice->setAdviceBody('verifUser');
    $this->addAdvice($verifAdvice);
  }
 public function
        verifUser(AOWP_JoinPoint $joinPoint)
        ... // omitted
 }
}
```

図 3 An Example Aspect

an underlying mechanism, and contains some of PHP specific data.

Aspect code in AOWP looks like the one in Figure 3. As seen from the example snippets, an AOWP aspect is introduced as a subclass of a pre-defined class constituting the framework. The aspect `UserRightAspect` is introduced as a subclass of `AOWP_PerSessionAspect` to make its own extent `PerSession`. Since an instance of `AOWP_RequestPointcut(...)` is created, its pointcut designator is of `Request` type to match with particular HTTP requests. The advice body `verifUser` is actually a `before` advice. The code also shows the advice function always takes a pre-defined argument of type `AOWP_JoinPoint`.

Weaving in AOWP is a batch process of source program translation. AOWP weaver accepts the base PHP source files together with those for aspect definitions, and generates the woven sources. The resultant PHP program is then loaded into Web server to start its execution to accept requests from remote clients.

2.3 Aspectual Encapsulation

AOWP has been applied successfully [8] [9] demonstrating the usefulness of the aspect in Web applications. Web-specific pointcuts such as `Request` and a variety of aspect extent are shown applicable well. Further, most of the aspects are homogeneous; the pointcut designators are chosen so that they can match with join points. For example, one to look at all the HTTP requests takes a form of

```
new AOWP_RequstPointcut('.*')
```

where the condition specified with a regular expression of `.*` stands for anything. In another example, a pointcut designator

```
new AOWP_RequstPointcut('index\.php')
```

looks for all the requests to invoke `index.php` page. It is less *homogeneous* than the first one since it matches only with requests to `index.php`. It, however, is still homogeneous in that the pointcut designator can select all the HTTP requests to have parameters such as

```
http://.../index.php?action=edit
```

Further, the following pointcut designator can be used in AOWP to match exactly with the above HTTP requests.

```
new AOWP_RequestPointcut('index\.php',
                    array('action' => 'edit'))
```

In the extremal case of heterogeneous aspect, one can be defined to match with a particular join point only. It seems not much different from a method called at a particular location; the join point at which the advice code is invoked is uniquely determined with the detailed pointcut. Owing to the obliviousness of AOP, the base program needs not know about the the advice code at all. It affects much on the style of software development.

図 4 Screen Snapshot

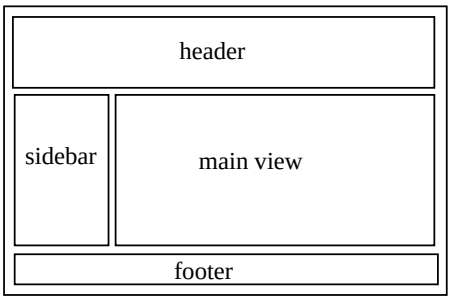

図 5 Screen Layout

3. Picture Management System

This section reports a Web application development practice to make fully use of the aspect-oriented programming techniques. In the development, AOWP is used to show that many interesting application features are encapsulated in either homogeneous or heterogeneous aspects.

3.1 Overview

Picture Management System (PMS for simplicity) is a Web application to manage picture data. A client is either a publisher or a subscriber. (S)he first sees the screen view shown in Figure 4, and usually connects to the system by following the authentication process. The system always checks to see if the client has a proper access right. A publisher can add or delete picture data, but a subscriber is allowed to see

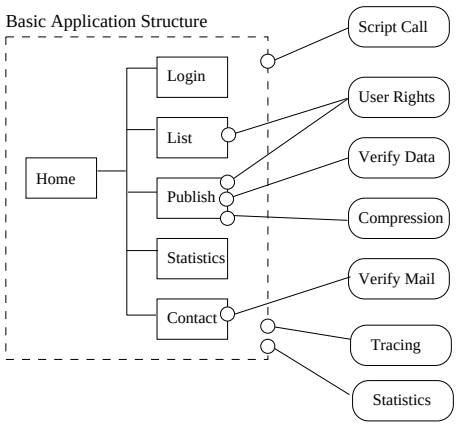

図 6 Feature Aspects Overview

the list of items and retrieve pictures only. While these are the basic features, PMS is desirable to provide further ones. Some of them are ready to be included and others are identified only after early versions of PMS are developed. Namely, they are introduced in an incremental manner. In developing PMS, most of the features are implemented with the notion of aspect, which demonstrates how AOWP supports the incremental development of Web applications.

3.2 Feature Aspects

Beside basic concerns such as the visual layout or HTML codes, PMS includes several concerns developed with AOWP aspects. Figure 6 gives an overview of the aspects and shows how they are related in the base system structure. In the figure, each box represents a basic application feature, which appear to the clients in the `main view` area of the screen layout illustrated in Figure 5. A list of menu is shown in the `header` area. Clicking one of the menu entries initiates a HTTP request to the Web server and the corresponding PHP program is invoked. As a response, `Main View` area is refreshed for a new sub-page to appear.

In Figure 6, each round box represents an *aspect*, which is shown to link with the base application whose pointcut is looking at. For example, `UserRights` aspect is woven to `List` base feature, and `Tracing` aspect is to many Web pages. In the following, each aspect is explained in detail.

3.2.1 Security Concerns

The security concern includes the one for authentication. It is the process of determining whether a client is a valid user or not. Further, it tries to prevent those unauthorized clients have access to some Web pages. In PMS, an unauthorized client cannot see the list of pictures.

a) User Rights Aspect

PMS has two profiles of users, publishers and subscribers. Only a subscriber is allowed to access Publish page, and a subscriber can view the list of items. When a user requests `Publish` page, this aspect checks the session's information: if the user is a publisher, (s)he will be able to add an item to PMS. If not, a pop-up is displayed and the user is redirected to Login page. The AOWP codes were shown in Figure 3.

3.2.2 Tracing Concerns

Web applications are desirable to include features to records all the user accesses. It is particularly important from an operation viewpoint since the traced data can provide valuable informations for the Web master. They represent users' scenarios visiting the Web site, and may help to re-consider the design of Web page transitions. In the current version of PMS, two such aspects `StatisticsAspect` and `TracingAspect` are introduced as AOWP aspects.

a) Tracing Aspect

`TracingAspect` is best understood as a well-known *logging* aspect, but puts its focus on collecting all Web pages visited by a particular user. This aspect is rather homogeneous in that it looks at all the HTTP requests. The extent of aspects implementing this concern would be `PerSession`.

```
class TracingAspect extends AOWP_PerSessionAspect
```

b) Statistics Aspect

`StatisticsAspects` records how many times a particular Web page is visited by all the users. It generates `Statistics` page for users to inspect. Since just one instance of this aspect is responsible for keeping track of all the information, the extent should be `PerApplication`.

```
class StatisticsAspect
                extends AOWP_PerApplicationAspect
```

3.2.3 Server-Side Validation Concerns

Web applications accept various information from users as data of String type, which are sometimes to be checked its well-formedness before handling them as the input data. Some check could be done at Web client side using such as JavaScript codes. It, however, is not always the case since there may be a chance for Web client not to execute such JavaScript codes or some check requires the information maintained at the server. As exemplars, PMS has two such aspects as described below.

a) Verify Data Aspect

`VerifyDataAspect` is responsible for the input validation to prevent the uses of non-supported characters when the input data are saved on database. Its point cut designator is associated with `insert` function and operates on each call of this. Actually, the aspect is instantiated before each call of `insert` function to check its parameters. Once it is finished, the aspect instance is destroyed. The aspect is so defined as `PerJoinPoint` with `FunctionCall` point cut.

```
class VerifyDataAspect
              extends AOWP_PerJoinPointAspect {
  public function _construct() {
    $dataAdvice = new AOWP_BeforeAdvice();
```

```
    $dataPC = new AOWP_FunctionCallPointcut
                                ('insert', 4);
    $dataAdvice->setPointcut($dataPC);
    $dataAdvice->setAdviceBody('dataControl');
    $this->addAdvice($dataAdvice);
  }
  ...
}
```

b) Verify Mail Aspect

Web applications generally have a particular Web page to allow users to send email to Web master. The page may provides a way to enter some information of the user himself including his own email address. `VerifyMailAspect` checks if the email format is correct, and further checks if its domain is valid. Depending on the error types, a particular alert page would appear to the user. Just as `VerifyDataAspect`, this aspect is defined as `PerJoinPoint` with `FunctionCall` point cut to match with each call of `mail` function.

```
$mailPC = new AOWP_FunctionCallPointcut('mail', 4);
$mailAdvice->setAdviceBody('mailControl');
```

3.2.4 Memory Consumption Concerns

Adding new items such as pictures or videos to the server may consume large amount of memory space. It usually requires to compress a large file before storing. PMS uses a mechanism of automatic compression of files before adding them to the server. Instead of implementing the feature in the base application program, an aspect is introduced to achieve it in PMS so that the algorithm of compression can easily be replaced. A new aspect to have a new algorithm is just woven without making any modification to the base program. In summary, the motivation of using AOP technique for this concern is a slight increase in its ease of modification.

a) Compression Aspect

`CompressionAspect` takes care of compression of large data before storing it in the database. In PMS, such storing is achieved with `deposit` function. The aspect looks only at calling this particular function, and thus is very heterogeneous. Namely, `CompressionAspect` matches only with the join point of calling `deposit`. The advice function `fileZip` may further call other function whose compression algorithm is meant for a specific file type. This aspect is defined as `PerJoinPoint` with `FunctionCall` point cut to match with each call of `deposit` function.

```
$ZipPC = new AOWP_FunctionCallPointcut('deposit', 1);
$ZipBefAdvice->setAdviceBody('fileZip');
```

3.2.5 Script Call Concerns

For several reasons, there might be some situations where a small script code executes outside the basic Web application to handle various exceptional cases. Such needs sometimes appear after completing an early version, and figuring out such cases systematically is not easy. A new feature comes up in an ad-hoc manner. In order to introduce such features as needed, an aspect is defined to include a feature of calling an external script code.

a) Alert Aspect

`AlertAspect` is developed for an instance of Script Call. It generates an HTML document to include a JavaScript code executing on the client Web browser. A particular example used currently in PMS is a JavaScript which notifies the user a time expiration alert. When a specified time is passed after the user log-outs from PMS, the code displays an alert and redirects him to `Login` sub-page. The extent of this aspect is `PerSession` because it is responsible for keeping track of a particular user. The aspect is heterogeneous only to look at `Logout` sub-page. It is a `before` advice since the responsibility of the code is just to insert the specified JavaScript code into the HTML document returned to the Web browser. The JavaScript pops up an alert window afterward.

```
$alertPC = new AOWP_RequestPointcut('index\.php',
                  array('page' => 'logout\.php'));
$alertAdvice = new AOWP_BeforeAdvice();
$alertAdvice->setAdviceBody('alertFunction');
```

4. Discussions

Aspect-Oriented Programming (AOP) [5] [11] is an alternative approach to achieving high modularity of program codes. Modularity is usually following hierarchical decomposition of system [14]. It sometimes needs concerns cross-cutting over the basic module structure or a primary concern. AOP provides a new language entity *aspect* to enable encapsulating such cross-cutting concerns. `UserRightAspect` is a typical example since implementing the access checking at runtime usually cross-cuts multiple Web pages.

Looking at a system from various concerns plays a key role in modeling at an early stage of the development. Such a modeling approach helps identifying new features by studying the system from multiple viewpoints [13]. In the case of PMS, `Tracing` is considered as one such example since the information collected with this aspect is meant to use by Web master. Its necessity is recognized when PMS is looked at from a viewpoint different from regular clients.

Further, FODA [10], a modeling method for Software Product-Line Engineering (SPLE) can be potentially related to AOP. A naive view is that common features constitute the primary concerns and each variability feature is mapped to an aspect. K. Lee et al [12] discuss how the feature-oriented analysis and programming in AspectJ are related. According to their experience, it is not always the case that every variability feature is mapped to an aspect. Some variability may be implemented by Java class as a part of the primary

concern.

In this paper, an alternative extremal approach is taken to use aspects in the development of PMS. The idea behind it is that the application feature written in PHP can be better separated from design of the Web page and Web page transitions than a monolithic PHP programs. Although a Web page can be considered to encapsulate PHP scripts and provides a basis for the page transition, much PHP script codes for realizing the application features reside in it. Such application features are desirable to be separated from the Web page skeleton, which may be achieved by AOP, especially with AOWP. To study this prospect further, the aspects, either homogeneous or heterogeneous, are extensively used in the development of PMS. It also demonstrates that AOP is useful in the incremental development style. Some of the aspects such as `Tracing` and `Alert` have been identified and introduced after early versions of PMS were developed.

New application features came up as the early versions were demonstrated. Conventionally, codes responsible for such new features are injected into the primary program base. Instead, an aspect was defined for each new feature regardless of the aspect being either homogeneous or heterogeneous. Adding a new feature is then just a weaving of new aspect with the basic PMS. It simplifies the process of validation or testing because the basic existing codes are not necessary to touch at all. What should be tested is the aspect only, eliminating some of the regression tests.

In general, a Web application system does not grow monotonically. Sometimes, an application feature may become obsolete and will not be used afterward. Such a feature should be removed from the system in order to keep the program size adequate. Obsolete or redundant codes may become a risk for security attacks since they are usually not maintained.

Removing an aspect is as easy as, or even easier than adding a new one. Thanks to the obliviousness that the base program codes know nothing about aspects woven into, discarding an aspect does not require any of the usual impact analysis [16]. The ease of removing also helps in debugging an aspect. If a particular aspect is being debugging, the other aspects could be unwoven temporarily. It should be noted here, however, that some aspects may depend on each other, which requires an impact analysis of aspects. A tool such as Celadon [18] for AspectJ could be applied to AOWP as well.

5. Conclusions

In this paper, we reported our experience in developing a Web application, in which we have made use of AOP technology extensively. In the development, we have used AOWP which is a new AOP framework for PHP programs. We have demonstrated that many interesting application features were encapsulated in either homogeneous or heterogeneous aspects. Although the discussion would be more like qualitative, the experience shows that the notion of aspect has a great advantage in an incremental style of Web application development.

Acknowledgments

The work reported here was conducted while Reda AHROUM (ENSIMAG, France) and Daniel BALOUEK (Universite de Pierre et Marie Curie, France) visited NII as their internship in the summer of 2009.

文　献

[1] AOWP : Aspect-Oriented Web Programming. http://posl.minnie.ai.kyutech.ac.jp/projects/aowp/.

[2] E. Andersson, P. Grenspun, and A. Grumet. *Software Engineering for Internet Applications*. The MIT Press 2006.

[3] S. Bergmann and G. Kniesel. GAP: Generic Aspects for PHP. In *Proc. EWAS'06*, 2006.

[4] A. Colyer and A. Clement. Large-scale AOSD for Middleware. In *Proc. AOSD'04*, pages 56–65, 2004.

[5] R.E. Filman, T. Elrad, S. Clarke and M. Aksit. *Aspect-Oriented Software Development*. Addison-Wesley 2005.

[6] R.E. Filman and D. P. Friedman. Aspect-Oriented Programming is Quantification and Obliviousness. in [5], pages 21–35, 2005.

[7] J. E. Garcia. Aspect-Oriented Web Development in PHP.

[8] K. Hokamura, N. Ubayashi, and S. Nakajima. Aspect-Oriented Programming for Web Controller Layer. In *Proc. APSEC 2008*, pages 529-536, 2008.

[9] K. Hokamura, R. Naruse, M. Shinozuka, N. Ubayashi, and S. Nakajima. AOWP: Web-specific AOP framework for PHP. In *Proc. ASE 2009 (Tool Demo)*, 2009

[10] K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson. Feature-Oriented Domain Analysis Feasibility Study. CMU/SEI-90-TR-21, 1990.

[11] G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-Oriented Programming. In *Proc. ECOOP'97*, pages 220–242, 1997.

[12] K. Lee, K.C. Kang, M. Kim, and S. Park. Combining Feature-Oriented Analysis and Aspect-Oriented Programming for Product Line Asset Development. In *Proc. SPLC'06*, 2006.

[13] B. Nuseibeh, J. Kramer, and A. Finkelstein. A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification. *IEEE Trans. Soft. Engin.*, Vol.20, No.10, pages 760–773, 1994.

[14] D.L. Parnas. On the Criteria to be Used in Decomposing Systems into Modules. *Comm. ACM*, Vol.15, No.12, pages 1053–1058, 1972.

[15] F. Ricca and P. Tonella. Analysis and Testing of Web Applications. In *Proc. 23rd ICSE*, pages 25–34, May 2001.

[16] I. Summerville. *Software Engineering (8th ed.)*. Addison Wesley 2007.

[17] J. Stamey, B. Saunders, and S. Blanchard. The Aspect-Oriented Web. In *Proc. SIGDOC'05*, pages 89–95, 2005.

[18] S. Zhang, Z. Gu, Y. Lin and J. Zhao. Celadon : A Change Impact Analysis Tool for Aspect-Oriented Programs. In *Proc. ICSE'08*. 2008.