ISSN: 2321-7782 (Online)
Volume 3, Issue 4, April 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Accumulation of Aspect- Oriented Features to MATLAB: Issues and Problems

Rekha Meena[1]
AIM & ACT,
Banasthali University,
Newai, India

Vaibhav Vyas[2]
AIM & ACT,
Banasthali University,
Newai, India

Abstract: ***Aspect-oriented programming permits software developers to supplement programs by means of information out of the scope of the base language while not hinder the code readability and also its portability. MATLAB is a well known modeling/programming language that can notably advantage from aspect-oriented programming quality. The Feature-oriented reengineering procedure offers strategy to construct a Software Product Line (SPL) out of legacy software benefit by means of feature model to hold up the formation of other development benefit, such as the product line architecture. This work suggests an aspect-oriented feature analysis to hold up the demonstration of both crosscutting concerns and variability, enabling to cause about how crosscutting concerns influence each other and other features and also to trace both variability and crosscutting concerns, which enables to measure the impact of, alter requests.***

Keywords: ***Aspect-Oriented Programming, MATLAB, fault tolerance assessment, advice, Feature-oriented reengineering process.***

I. INTRODUCTION

MATLAB is an elucidated, very important programming language planned to function mainly on matrix-shaped double precision data types. Accessible MATLAB character and collection assist programmers to concentrate on problem solving and permit high expressiveness when marketing with matrix computations, thus granting to improved yield. It is broadly utilized in signal processing, scientific computing, control systems, system engineering, image processing and simulation. MATLAB depends a lot on array variables and double precision data types. At each instance fresh characters are required, insidious alterations on the existing key are necessary, as well as the introduction of new key. These issues are felt in affairs associated to implementation standard, like those emerging in embedded system applications as MATLAB can be approached as a specification, rather than as an implementation language. AOP benefits the abstraction of security-related programming tasks such as authentication, access control and integrity is the basic advantage of using AOP within security. These security concerns are likely to crosscut objects. Crosscutting concerns are related problems that are spotted all through the functionality of an application. Security aspect may be used again for other applications are an extra advantage [1].

A. Overview of AOP

AOP provides a graceful approach to introducing faults and assertions into original code in every place that fault tolerance requires to be tested, without altering the existing code of the program. We used AspectJ, an aspect-oriented extension to Java, which offers the following [2]:

1) *Upward compatibility :* Every ligitimate Java programs are ligitimate AspectJ programs.
2) *Platform compatibility :* Every ligitimate AspectJ programs run on standard Java virtual machines.
3) *Tool compatibility :* AspectJ works as an extension to original tools, counting integrated development environments (IDEs), documentation tools, and design tools.

4) *Programmer compatibility :* Programming with AspectJ feels like a natural supplement of programming with Java.

B. Aspect-Oriented Programming

AOP recounts a programming entity particularly blueprint to deal with the execution of crosscutting concerns– concerns that, although theoretically different, partly cover with the execution of other concerns in the code by contribution the identical functions or classes. In AOP, aspects encapsulate parts of code is known as advice which execute a crosscutting concern as a separate module. A part of advice preys several of join points narrated by a predicate called pointcut expression. Pointcut expressions are assessed by the aspect weaver that weaves the code from the advice framework to the join points which are coordinated by the individual predicates [3].

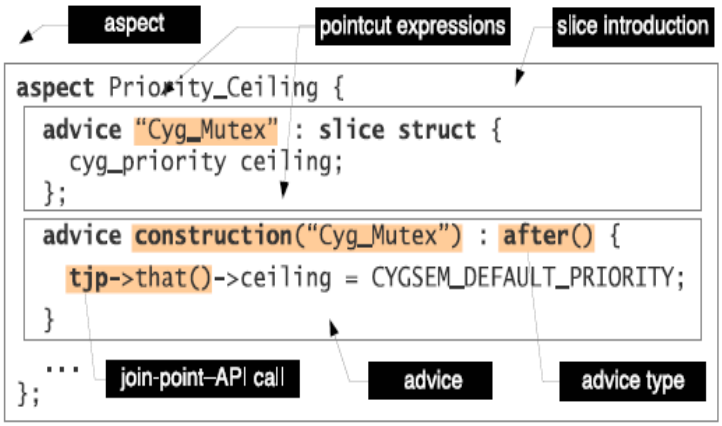

Figure1: Syntactical elements of an aspect

II. LITERATURE REVIEW

According to João M. P. Cardoso et al. [1] MATLAB is an elucidated, very important programming language planned to job mainly on matrix-shaped double precision data types. Paper presents a view point for stating modifications of MATLAB programs in an aspect-oriented manner, with a concentrate on optimization concerns. They described a set of characteristics for a domain-particular language to program approaches, structured as aspect modules. Maintaining a severe division between a MATLAB bottom and fresh aspect-oriented characters gifts to better maintenance, readability, and use again both the base programs and aspects.

Kung Chena et al. says that [4] access control logic is separated from the centre of application and gathered into individual aspect modules that are synthesized mechanically from access control policy in XML format and correctly planned aspect templates. We sketch to plan and put into practice a rule development device that aids security administrator translate the abstract access control policy into the XML format. Once the numbers of rules rise to a extent it is very probable that we will accidentally generate a few contradiction between associated rules that may slide into the aspect code lacking observation.

According to Leonardo P. Tizzei et al. [5] the Feature-oriented reengineering process offers guidelines to build a Software Product Line (SPL) from legacy software assets by means of feature model to hold up the formation of other development assets, like the product line architecture. The main involvement of this work is a new feature view that supports modelling crosscutting concerns and variability by means of a notation alike to the feature model.

III. Proposed System Implementation Issues Of MATLAB

Aspect modules schemes and MATLAB key are precised in individual origin document. A front-end parses the insert MATLAB key and transforms the gained abstract-syntax tree into a particular IR. Tool used is TOM, a high-level program rewriting structure. TOM receives the description of the policy and the rewriting scheme and involves a design matching engine. Tags entrenched in MATLAB key to state joinpoints are organised and entrenched in the acquired IR and approved in this type by the MATLAB compiler front-end to the other tools in the compilation flow. Symbol tables to the tools as a data types and shapes are made accessible as in the compilation flow. A transformation engine cavort the task of aspect weaver, gaining the IR as input and making a changed IR which encompasses the characteristics specified by the aspect modules. The weaver is being executed by means of the entity of strategic programming as offered by TOM. It finds out the orders of aspects to carry out based on the aspect schemes. Other concerns, like observing and key alterations, are also composed to the IR of the original MATLAB program through the weaver, which gives a changed IR made available to the successive tools in the compilation flow. Each one is essential for dissimilar sides of the process. Code generation also takes benefit of the TOM code rewriting abilities [1].

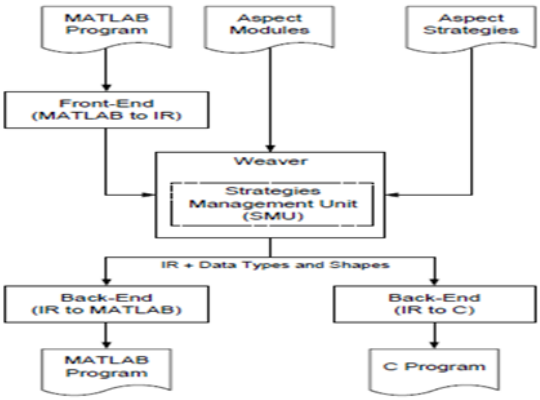

Figure2: Environment under development

It is not simple to build up a comprehensive however flexible mechanism for access control in HIS with EMR (electronic medical record). There are minimum two foremost problems. First, as other certainty needs, access control is a throughout system issue that pervades via all the main components of a scheme. This is not only error-prone but also makes it hard to confirm its accuracy and carry out the essential preservations. Second, access control rules in healthcare domain are inherently fine-grained and active. It is ordinary for developers of information system to divide users into dissimilar groups and specify access affluent in terms of the application jobs that a specific class of users has a right to access. For HIS (healthcare information systems), however, an extra level of access control must be clear at the *data field* level. Access may be restricted to particular patient records or specific elements within a patient record. For example, while a physician can see a few fields of a patient's EMR, only the patient's *attending* physician can observe the complete documentation and alter it. On the other side, we will have to bypass the limitations in an extremity. Additionally, modification in jurisprudence or modification in the prediction of jurisprudence can result into main emendation of the access control schemes. All these results into the requirements of frequent changes for access control schemes in healthcare domain [4].

A. Why AOP is a Promising Solution

There are numerous characteristics inbuilt in AOP that are gifted for controlling the disadvantage [3].

» Firstly, AOP introduces a latest type of fastening between modules. The advice-based fastening method provided by AOP can efficiently reverse that association. The callee can join together itself into the caller without the caller having to be familiar with the callee.

» Secondly, by providing the method of quantification through pointcut-expression matching, one of major demonstrate motivation of AOP is that it permits for a modularized execution of crosscutting concerns.

IV. AOP For Fault Tolerance Assessment

Aspect-Oriented Programming is a programming method that enables the requirement of cross-cutting aspects in high-level languages that get woven into source code at compilation or run time. In other words, at the same time traditional languages, like Java, hold up constructing a program around a dominant disintegration, like the functional classes building up an application's business logic, aspects facilitate the separate programming of other aspects of a program's behaviour – like logging, synchronization, or quality of service (QoS) – that would have to be scattered all the way through the functional classes. An aspect weaver that works as individual stage of the compilation process mechanically inserts the aspect code in the suitable spaces all the way through the functional application [2].

A. Advantages of Using AOP for Fault Tolerance Assessment

» First, using AOP, and AspectJ in particular, enables the quick development of a library of faults for testing a system and addition of those faults into an original system [2].

» Second, we can weave the dissimilar types of faults required to completely consider fault tolerance into the dissimilar spaces in that they can take place in a scheme. Presently, the outline hold up the following kinds of faults[2]:

- *Direct faults* that take instant effect when triggered.
- *Conditional faults* which, when triggered, can take effect a little in the future, randomly, probabilistically and repeatedly.

» Third, using AOP means that the outline is extensible to append recent fault kinds, assertions, and injection points.

» Fourth, logging and metric collections are canonical use cases for AOP. An integral part of assessing fault tolerance is being able to observe system behaviour in response to fault injection [2].

» Fifth, using AOP maintains a clear division between the test code and the production.

V. Background: Feature-Oriented Software Development

Feature-oriented software development (FOSD) is a concept for the customization, establishment and synthesis of wide-reaching software systems. Feature modelling enables the illustration of the commonalities and variabilities of a software system. Because alike products can be derived from a SPL, feature modelling is generally used to symbolize mandatory, optional and alternative features of a SPL. Since feature recycle is one of FOSD aims, the Feature-oriented reengineering is a method to construct software product lines from bequest applications. For the sake of clarity, some assets were omitted. First, the legacy architecture is recovered from existing modules. Based on the recovered architecture and domain knowledge, properties are recognized and the feature model is produced. The subsequent activity is the *Feature model refinement*, which is influenced by market requirements and recent technologies which may anticipate quality evolution of applications. After the refinement of the feature model, product line architecture is designed and, finally, its architectural elements are implemented [5].

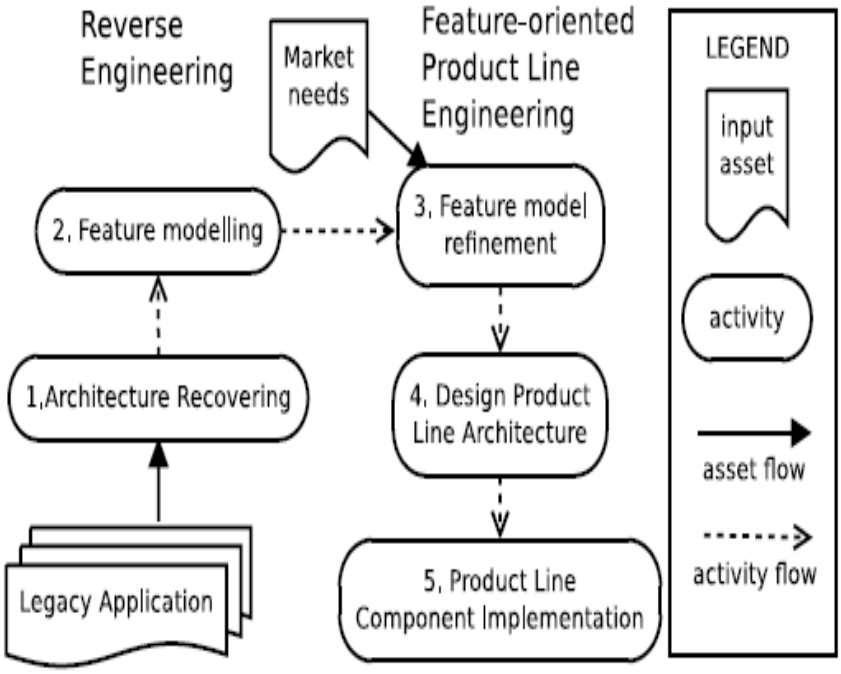

Figure3: A Feature-oriented reengineering process

A. Aspect-Oriented Feature View

1) Interplay between FOSD and AOSD:

TABLE I.

Definitions of Concepts from FOSD and AOSD

S.R.No.	Term	Definition
1.	Feature	• It is a prominent or distinctive user-visible aspect quality, or characteristic of a software system. • It is a distinguishable characteristic of a concept that is relevant to some stakeholder of the concept.
2.	Concern	• It is anything a stakeholder may want to consider as a conceptual unit, including features, non-functional requirements, and design idioms. • It is an area of interest or focus in a system, such as a requirement, a feature or a use case.
3.	Crosscutting Concern	• It is a concern that cuts across other concerns. These concerns are responsible for producing tangled representations that are difficult to understand and maintain. • It is a behaviour that cannot be encapsulated because of its impact across the whole system is called crosscutting behaviour.
4.	Aspect	• It is a module that is potentially able to encapsulate software or design artefacts treating an otherwise crosscutting concern. • It is a separate and independent module that modularises crosscutting concerns.

2) Mobile phone: an Illustrative example :

The *Aspect-oriented feature analysis* hold up the definition of crosscutting properties and their relationship with other properties and with each other.

This analysis has two main objectives:

(i) to state which properties are affected by each crosscutting properties, and

(ii) To recognize feasible conflicts surrounded by crosscutting features and help to answer them by prioritizing a few properties. The crosscutting feature view also supports SPL variability, since property kind may influence its relationships. For instance, a mandatory crosscutting property should not crosscut an elective property unless it also crosscuts other non-optional properties [5].

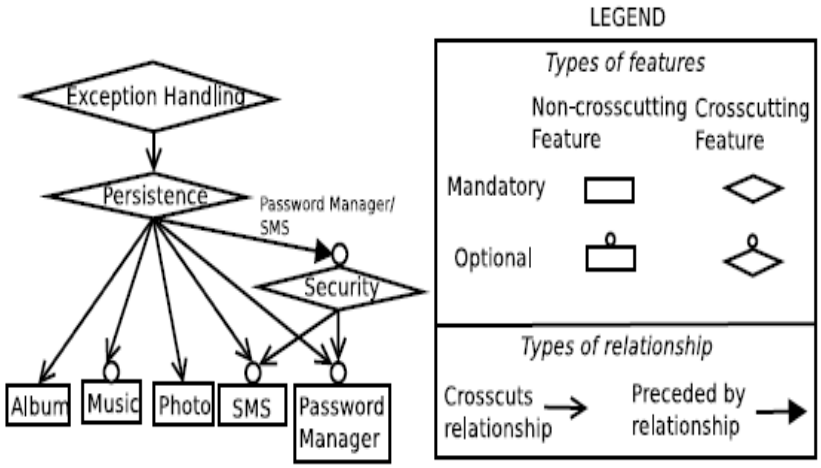

Figure 4: Aspect-oriented feature view

3) Formal specification of Aspect-oriented Feature View:

Here we set up formal definitions to accurately identify the *Aspect-oriented Feature View*. By offering a formal requirement we mean to illustrate the limitations and capabilities of the aspect-oriented feature view. This requirement is also an original footstep in the direction of the progress of a software tool. First we explain the composition of the outlook and its elements, subsets and sets, that are involved in the specifications associated to the *Aspect-oriented Feature View [5]*.

VI. CONCLUSION

Our paper presents a method for stating modifications of MATLAB programs in an aspect-oriented fashion, with centre on optimization issues. Charging a severe partition between a MATLAB base and new aspect-oriented properties grants to better maintenance, readability, and recycle of both aspects and base programs. We have provided an aspect-oriented method to provide adjustable access control for Electronic Medical Records on Web-based systems. Present aspect-oriented models can be used in the context of feature-oriented reengineering process to hold up design and implementation phases. The work presents an *Aspect-oriented feature view*, which models crosscutting concerns as crosscutting characteristics thereby enabling to reason about their impact on each other and on other non-crosscutting characteristics. The job of this sight is not to substitute the characteristic model, but to complement it.

References

1. João M. P. Cardoso, Pedro C. Diniz, Miguel P. Monteiro, João M. Fernandes and João Saraiva “A Domain-Specific Aspect Language for Transforming MATLAB Programs”, Portugal, 2010.
2. Jeffrey Cleveland, Joseph Loyall, James Hanna “An Aspect-Oriented Approach to Assessing Fault Tolerance” USA, 2014.
3. Daniel Lohmann, Wanja Hofer, Wolfgang Schröder-Preikschat, Jochen Streicher, Olaf Spinczyk “CiAO: An Aspect-Oriented Operating-System Family for Resource-Constrained Embedded Systems”, 2009.
4. Kung Chena, Yuan-Chun Changa, and Da-Wei Wangb “Adaptable Access Control for Electronic Medical Records”, 2006.
5. Leonardo P. Tizzei, Jaejoon Lee, Cecília M.F. Rubira “An Aspect-oriented View to Support Feature-oriented Reengineering”, 2010.

