

Analysing Requirements Dependencies and Change Impact using Concern Slicing

Safoora Shakil Khan and Awais Rashid

Computing Department, Lancaster University
{shakilkh, marash}@comp.lancs.ac.uk

Abstract. Identification of dependencies among requirements helps in predicting the scope (intensity and depth) of change impact. In this paper we propose a multidimensional concern-slicing approach that facilitates understanding of requirements dependencies and impact of requirements change. The concerns are sliced as temporal, conditional, task-oriented, and business rules. Based on stakeholders' preferences and system requirements the four concern slices are assigned to particular weight bands. Dependency graphs for each concern slice are constructed from semi-formal dependency equations, which assist in analyzing the consequential change impact.

1 Introduction

Requirements have a tendency to evolve during system development and its lifecycle. Such evolution may be passive or active. Passive evolution refers to changes in system requirements in a time frame of a couple of years, decade, or more, e.g., when a client signs up for mortgage, till the mortgage period is over, the criteria for paying off mortgage remains the same. In contrast, in active evolution the system requirements change frequently over a short span of time, i.e., a couple of weeks or months, e.g., introducing new loan schemes, variations in account policies in a banking system. Evolution whether active or passive may lead to inconsistencies, therefore, it is necessary to identify dependency among requirements relationships to understand the probable impact of change. The understanding of requirements dependencies also helps in better management and maintenance of requirements.

In this paper we propose a multi-dimensional approach for requirements dependency management. The concerns are sliced as: temporal, conditional, task-oriented, and business rules. These slices are inspected as independent modules to identify forward, backward, and parallel intra (within the same concern) or inter (across different concerns) dependencies, which in turn helps in predicting the level of change propagation. Our approach determines the intensity and level to which change is likely to propagate.

The rest of the paper is structured as follows. Section 2 discusses our slicing of requirements level concerns. Section 3 discusses how dependency equations are derived and weights assigned to the concern slices. In Section 4 probable impact of

change is analyzed. Section 5 discusses the related work while section 6 states the prospective future directions and concludes the paper.

2 Slicing Requirements-Level Concerns

We have adapted the notion of program slicing by Weiser [1] to capture the dependencies among requirements-level concerns. We categorize a concern's requirements in to four slices: temporal, conditional, task-oriented, and business rule. Each slice has varying significance according to stakeholders' preference and system requirements. Slicing facilitates in understanding dependencies of each concerns' requirement on other concern slices. There could be more concern slices but we are just considering the four rudimentary slices to explain our approach.

In our approach the concerns are symmetrically treated, i.e., they are not restricted by the concept of two-dimensionality, i.e., functional concerns (base) and non-functional concerns (crosscutting) [2]. Therefore the elicited concerns are flexible to act as either base or crosscutting depending on the compositional needs [3, 4]. This multidimensional characteristic of our approach enables us to treat the elicited concerns as multifaceted (cf. Fig.1).

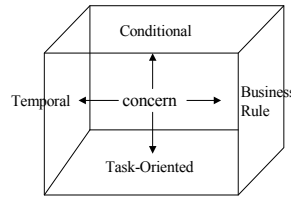


Fig.1. Multifaceted concerns in requirements

For the purpose of explaining our approach we use the Portuguese tollgate case study from [5] in which the owners of authorized vehicles have to register the vehicle with the bank and activate a gizmo by using an ATM. When the owner uses the green lane on the motorway the sensors detect and read the gizmo automatically deducting toll amount based on the type of tollgate, type of vehicle, and the distance traveled. Due to space limitation we will consider only concerns: register, activate, and record (cf. Fig.2). The concern slices are shown as S_T : Temporal, S_C : Conditional, S_{TO} : Task-Oriented, and S_B : Business rules. Each concern's requirements are categorized into its respective slices as follows:

Temporal Slices: A requirement is framed into a temporal slice if it shows timeliness property. For example, the gizmo is read by tollgate sensor in time t .

Conditional Slices: A requirement is framed into a conditional slice if it is dependant on execution of a particular criterion to be fulfilled. Till the time the criterion is not fulfilled the concern is in block mode, i.e., the next step is not executed till the condition is met. For example (cf. Fig 2.b.) requirement R1 of *record*

concern is categorized as conditional concern because the gizmo needs to be identified before the information in the gizmo is stored by the system. Similarly (cf. Fig 2.c.), requirement R3 of *activate* concern is categorized as a conditional requirement because the ATM will not inform the system regarding gizmo activation till step R2 has not been executed, i.e., gizmo activated using ATM.

Business Rule Slices: A requirement is framed into a business rule slice if it describes organizational policies, decisions, structures, etc. For example (cf. Fig. 2.a.), requirement R1 of *register* concern is categorized as business rule because new fields of information may be introduced on changes to the organization’s business policies. Similarly (cf. Fig.2.c.) requirements R1, R2, and R3 of *activate* concern are categorized into the business rule slice because we may change R1 so that the client is not sent the gizmo for activation, but activated by seller on purchase, for R2 and R3 the mode of activation may change from ATM to other mode, i.e., activated online.

Task Oriented Slices: A requirements is framed as task-oriented if it is dependant on external input or response from the user. For example (cf. Fig 2.c.), requirements R1 and R2 of *activate* concern are categorized into the task oriented slice because they are dependant on an input, response, or an external activity to occur.

<p>Concern Name: Register Concern Number: C1 <i>Business rules Slice (S_B):</i> R1: Registration of authorized vehicles includes the owner’s personal data, bank account number, and vehicle details. <i>Conditional Slice (SC), Task-oriented Slice (STO), and Temporal Slice (ST):</i> No requirements</p>	<p>Concern Name: Activate Concern Number: C2 <i>Temporal Slice (ST):</i> No requirements <i>Task-oriented Slice (STO):</i> R1: Gizmo sent to client to be activated R2: Gizmo activated using ATM <i>Business rules Slice (SB):</i> R1: Gizmo sent to client to be activated R2: Gizmo activated using ATM R3: ATM informs system regarding gizmo activation <i>Conditional Slice (SC):</i> R3: ATM informs system regarding gizmo activation</p>
<p>Concern Name: Record Concern Number: C3 <i>Conditional Slice (S_C):</i> R1: The information read is stored by the system <i>Task-oriented Slice (S_{TO}), Business rules Slice (S_B), and Temporal Slice (S_T):</i> No requirements</p>	<p style="text-align: right;">c.</p>

Fig. 2.a. Register concern and its slices b. Record concern and its slices c. Activate concern and its slices

Fig 2. shows that a concern can comprise of multiple slices or at least one slice. Each slice comprises of concerns’ requirements and these requirements can be categorized to multiple slices, e.g., requirement R2 of activate concern is categorized as both task oriented slice and business rule (cf. Fig. 2.c.).

3 Dependency Equations

The dependencies among the concern slices can be: intra or inter dependant. A concern slice depending on a slice within the same concern is termed as intra concern dependency. An inter concern dependency is when a slice depends on another concern's slice. The intra and inter dependencies can be forward, backward, or parallel.

Forward Dependency: if a concern slice links/results to another concern slice, then the former concern slice is said to be forward dependant on later, e.g., in the *activate* concern task oriented and business rule slice $S_{TO^*B}(R2)$: 'gizmo activated using ATM' and $S_{C^*B}(R3)$: 'ATM informs system regarding gizmo activation' are linked in a sequence to each other. Hence it can be termed as $S_{TO^*B}(R2)$ is *forward* dependant on $S_{C^*B}(R3)$

Backward Dependency: if a concern slice uses or bases itself on the previous concern it is termed as backward dependent e.g., *activate* concern slice $S_{C^*B}(R3)$ is *backward* dependant on *activate* concern slice $S_{TO^*B}(R2)$. Until the gizmo is activated the ATM will not inform the tollgate system regarding gizmo activation. This shows that execution of a previous task is mandatory for execution of next task.

Parallel Dependency: Two concern slices occurring concurrently are termed as parallel dependant concern slices', e.g., *compatibility* concern's business rule slice $S_B(R1)$ 'ATM is compatible with the system and vice versa' must exist/occur in *parallel* with *activate* concern slice $S_{C^*B}(R3)$.

The requirement analyst observes the slices and concerns requirements to identify dependencies among them to derive semi-formal dependency equations. To express the dependency as semi-formal equations we define three dependency keywords:

Links: a semi-formal interpretation for forward dependency, e.g., C1 *links* to C2, means that C1 is forward dependant on C2.

Uses: a semi-formal interpretation for backward dependency, e.g., C4 *uses* C2 means that C4 is backward dependant on C2.

Occurs: a semi-formal interpretation for parallel dependency, e.g., C4 *occurs* with C10, means that C4 is parallel dependant on C10.

Dependency equation for *register* concern can be formulated as follows:

$$C1.S_B(R1): \textit{links} \text{ to } C3.S_C(R1) \quad (1)$$

Equation (1) shows that *register* concern's business rule slice is forward interdependent on *record* concern's conditional slice $C3.S_C(R1)$.

$$C2.S_{TO^*B}(R1): \textit{links} \text{ to } C2.S_{TO^*B}(R2) \quad (2)$$

Equation (2) shows that *activate* concern's task-oriented and business rule slice is forward intra dependent on $S_{TO^*B}(R2)$.

$$\mathbf{C2.S_{TO^*B}(R2)}: \text{links to } \mathbf{C2.S_C(R3)} \wedge \text{uses } \mathbf{C2.S_{TO^*B}(R1)} \wedge \text{occurs with } \mathbf{C6.S_T(R1)} \quad (3)$$

Equation (3) shows that *activate* concern's task-oriented and business rule slice $S_{TO^*B}(R2)$ is forward intra dependant on concern slice $S_C(R3)$, backward intra dependant on $S_{TO^*B}(R1)$, and parallel interdependent on the *response* concern's slice $C6.S_T(R1)$.

$$\mathbf{C2.S_{C^*B}(R3)}: \text{uses } \mathbf{C2.S_{TO^*B}(R2)} \wedge \text{occurs with } \{\mathbf{C14.S_{TO^*B}(R1)}, \mathbf{C10.S_B(R1)}, \mathbf{C6.S_T(R2)}\} \quad (4)$$

Equation (4) shows that *activate* concern $S_{C^*B}(R3)$ is backward intra dependent on concern slice $S_{TO^*B}(R2)$ and parallel interdependent on *multi-access* concern $C14.S_{TO^*B}(R1)$, *compatibility* concern $C10.S_B(R1)$, and *response* concern slices $C6.S_T(R2)$.

After formulating the dependency equations we assign weights to these equations. The weight assignment helps in assessing the scope (intensity, level, and importance) of change impact on the concerns. The weight assignment criterion depends on the context, preference, and system quality requirements. For example in the tollgate system business rule slices, temporal slices, and conditional slices are of significant importance. Business rule concern slices are of highest significance because if any business rule is changed it will impact the entire system, e.g., if mode of gizmo activation is changed from ATM to online gizmo activation the change will impact all the system concerns. Similarly, the introduction of a new policy, e.g., fluctuation in toll deduction policies will affect all the associated concern slices. Therefore a business rule is considered to be of critical importance and assigned the highest weight band. Temporal slices and conditional slices are of similar significance on the basis that if there is a change in time in one of the slices it will propagate to all the concern slices and if a condition is not met then the next step will not execute. For example, the timing to turn on yellow light is changed from 0.2 sec to 0.9 sec. The change in the timing will then propagate to the action of photographing the number plate and fine calculation for the unauthorized vehicle. Similarly if the sensor does not identify the gizmo, the corresponding conditional steps: toggling the state of light (green or yellow), toll calculation, photographing the vehicle's number plate, or fine calculation are not executed. The tollgate system is a real time system and requires optimum performance in order to facilitate the motorway users. Therefore temporal slices and conditional slices have the same weights.

The weight criteria for concern slices for tollgate system case study are:

- Task-Oriented Slice: [0.1 to 0.3]
- Conditional Slice: [0.5 to 0.7]
- Temporal Slice: [0.5 to 0.7]
- Business Rule Slice: [0.8 to 1.0]

Each weight band has three values to define the intensity of the concern slice with respect to the dependant slices. The intensity range is *low*, *mild*, and *severe*.

To explain the mechanism of weight assignment for concern slices based on dependencies we assign weights to our derived dependency equations (1) and (3).

First let us consider equation (1) *register* concern's business rule slice **C1.S_B(R1)**:

- Forward interdependent with *record* concern's conditional slice **C3.S_C(R1)**

Now let us analyze the dependencies for **C3.S_C(R1)** to determine the change impact at second level:

- Forward interdependent on *debit* concern's business rule and conditional slice **C5.S_{B^C}(R1)**: 'Toll amount or fine is debited from the owner of charged vehicle's account'.

Now let us analyze the dependencies for **C5.S_{B^C}(R1)** to determine the change impact at third level:

- Backward interdependent on *record* concern's conditional slice **C3.S_C(R1)** and *calculate* concern's business rule and conditional slice **C8.S_{B^C}(R1: R3)**
- Parallel interdependent with *correctness* concern's slice **C11.S_C(R7)** and *compatibility* concern's slice **C10.S_B(R2)**.

The dependency between **C1.S_B(R1)** and **C3.S_C(R1)** is assigned value 0.7 (severe intensity). Following are the reasons for severe change impact intensity calculated for **C1.S_B(R1)** which are based on dependencies observation and understanding (cf. Fig. 3.a. for pictorial understanding):

- a. **C3.S_C(R1)** is a conditional slice. Change in the business rule of **C1.S_B(R1)** will impact *record* concern's **C3.S_C(R1)** which will consequentially impact *debit* concern **C5.S_{B^C}(R1)**.
- b. The information read off gizmo, **C3.S_C(R1)**, helps system to identify the account to debit money from, this show forward dependency between the concerns. Therefore the forward dependency link between **C5.S_{B^C}(R1)** and **C3.S_C(R1)** is assigned value 0.6 (mild *conditional* intensity).
- c. The compatibility property states that both systems should be compatible with each other. Since the *compatibility* concern **C10.S_B(R2)** is not directly impacted due to change in **C1.S_B(R1)** they are not assigned any value (cf. Fig.3.a.)

Now let us consider equation (3) *activate* concern's business rule and task-oriented slice **C2.S_{TO^B}(R2)**:

- Forward intra dependant on **C2.S_C(R3)** states 'Gizmo read in time 't₂' by the tollgate sensor'..
- Parallel interdependent on *response* concern's conditional slice **C6.S_T(R1)**

Now let us analyze **C2.S_C(R3)** to determine the propagation of change to second level:

- Parallel dependant on *multi-access* concern **C14.S_{TO^B}(R1)**, *response* concern's slice **C6.S_{T^C}(R2)**, and *compatibility* concern's slice **C10.S_B(R1)**. The dependency link between **C2.S_C(R3)** and **C14.S_{TO^B}(R1)** is assigned an overall intensity 1.2, i.e., sum of 0.3 (severe *task oriented* intensity) and 0.9 (mild *business rule* intensity). The dependency link between **C2.S_C(R3)** and **C6.S_{T^C}(R2)** is assigned an overall intensity 1.2, i.e., sum of 0.5 (low *temporal* intensity) and 0.6 (*conditional* intensity) because the activation information can be forwarded after a couple of hours of activation as no

other concern is directly forward dependant on it or halts a time dependant execution. $C6.S_{TC}(R2)$ is backward intra dependant on $C6.S_T(R1)$, which is not assigned any value as change to $C6.S_{TC}(R2)$ will not affect predecessor concerns. The dependency link between $C2.S_C(R3)$ and $C10.S_B(R1)$ is assigned an overall intensity 1.0, i.e., (severe *business rule* intensity).

Now let us analyze $C6.S_T(R1)$ to determine the propagation of change to second level:





- Forward intra dependant on $C6.S_{TC}(R2)$ and dependency intensity is of 1.2 (0.6 mild *conditional* intensity and 0.6 mild *temporal* intensity). Conditional and temporal intensity is 0.6 because only $C6.S_{TC}(R2)$ is affected by change in $C6.S_T(R1)$.

From above dependency understanding and reasoning $C2.S_C(R3)$ is assigned weight 0.7 (severe conditional intensity) and $C6.S_T(R1)$ is assigned weight 0.5 (low temporal intensity).

4 Analyzing the Change Impact

After deriving the dependency equations and assigning them weight we construct dependency graphs for visual representation. These graphs help to visually analyze the level of change propagation due to concern slice modification and capture other semantic dependencies, which may not necessarily be sequence of events or state transitions. In other words dependency graphs help analyst understand the change impact and capture semantic dependencies. The visual representations for the dependency graphs are shown in Table 1.

Table 1. The visual representation for the dependencies and concern slices

Dependency	Visual Representation
Forward Dependency	
Backward Dependency	
Parallel Dependency	
Concern Slice under consideration	

On the basis of the formulated dependency equations we construct dependency graphs. Due to space limitation dependency graphs for Equation (1) and (3) are constructed (cf. Fig. 3.a. and 3.b.)

Let us analyze dependency graph in Fig. 3.a., which shows $C1.S_B(R1)$ affects $C3.S_C(R1)$ severely as change in business policy may require insertion of new data field or deletion of the existing data field. Due to this change the sensors at the tollgate system have to cater for the change in information fields. $C3.S_C(R1)$ affects the *debit* concern's conditional and business rule slice $C5.S_{CB}(R1)$ mildly. The owner

account details are provided from $C1.S_B(R1)$ which are recorded by *record* concern's slice $C3.S_C(R1)$. Hence change in $C1.S_B(R1)$ impacts $C3.S_C(R1)$ severely which has an indirect consequential impact on $C5.S_{C^B}(R1)$. The parallel dependencies $C11.S_C(R7)$ and $C10.S_B(R2)$ are not assigned weights as they occur concurrently with *debit* concern and not directly impacted by change in $C1.S_B(R1)$. Backward dependence $C8.S_{B^C}(R1:R3)$ is not assigned values as the *correctness* in toll amount calculation is not impacted by change to *record* concern slices. Backward dependency from $C5.S_{B^C}(R1)$ to $C3.S_C(R1)$ is assigned a value 0.6 (mild *conditional* intensity) as change in debit account number may provoke account record, type, or details to be changed in *gizmo*.

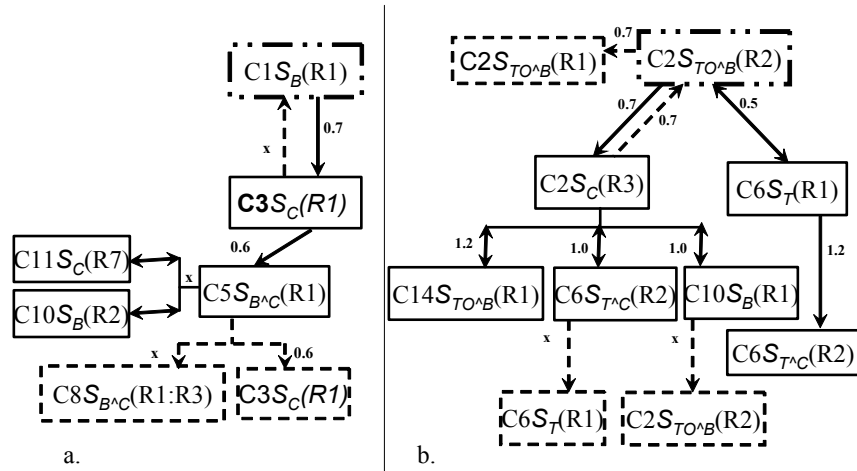


Fig. 3.a. Dependency graph for *register* concern's business rule slice $S_B(R1)$ **b.** Dependency graph for *activate* concern's task-oriented and business rule slice $S_{TO^B}(R1)$

The dependency graph (Fig.3.b.) visually represents dependency equation (3). It shows that $C2.S_{TO^B}(R2)$ is forward intra dependant on $C2.S_C(R3)$ and parallel interdependent on *response* concern slice $C6.S_T(R1)$.

First let us consider the $C2.S_C(R3)$ concern slice, which has severe intensity impact (0.7) and analyse its dependencies: From equation (4) we can observe that $C2.S_C(R3)$ is parallel dependant on multi-access $C14.S_{TO^B}(R1)$ concern slices with an impact intensity of 1.2, response $C6.S_T(R2)$ with an impact intensity of 1.2, and compatibility $C10.S_B(R1)$ with an impact intensity of 1.0. While $C6.S_{T^C}(R2)$ and $C10.S_B(R1)$ are backward dependant on $C6.S_T(R1)$ and $C2.S_{TO^B}(R2)$, respectively. Since it is a condition that has already occurred and the change to predecessor concern does not impact backward on these two concerns therefore they are not assigned any values.

Now let us consider the $C6.S_T(R1)$ *response* concern slice, which has comparatively low impact intensity (0.5) and analyse its dependencies: $C6.S_T(R1)$ has a severe change impact (1.2) on $C6.S_{T^C}(R2)$.

The dependency graphs in Fig.3.a. and Fig.3.b. enable us to identify the level and intensity of change propagation. From Fig.3.a. we analyse that if modification is made

to $C1.S_B(R1)$ it affects to two levels, i.e., $C3.S_C(R1)$ and $C5.S_{B^*C}(R1)$. From Fig. 3.b. we analysed that change to $C2.S_{TO^*B}(R2)$ will be confined to the *activate* concern's slices $C2.Sc(R3)$ and response concern's slice $C6.S_T(R1)$, which consequently have a severe affect on $C6.S^*C_T(R2)$.

5 Related Work

There has been very little work done for impact analysis techniques for aspect-oriented software. One of the significant works is [6] categorizes concerns as enduring (stable and related to system's domain) and volatile (modifiable). The business rules of an organization are likely candidates to evolve. [6] does not analyse the impact of change on dependant concerns. In comparison our approach breaks each concern into four slices: temporal, conditional, task-oriented, and business rule slices and each of these four concern slices have different impact intensity depending on the stakeholders' preference and system requirements. In addition our approach identifies the level and intensity of impact on the concern slices due to change based on the dependency equations, weights, and graphs.

Most of the impact analysis techniques for object oriented software focus on changes at the code/implementation level [7]. Our approach identifies the impact of change at the early phase of software development lifecycle, which helps in understanding the impact level and intensity of a change. Our approach adapts the program slicing concepts from [8, 9]. [8] slices program into decomposition and complement slice. Decomposition slice is isolated from the rest of the code on the basis of variable definition and use (defuse) to identify modifiable codes. Similarly our approach considers concern as a chunk comprising of four slices and identifies the impact of change on each concern slice. In [9] concern call graphs are created, which document the relationships between the various parts of concerns. Our approach is inspired from few of these code level impact analysis and maintenance techniques and uses inter and intra concern dependency graphs for a better understanding of concern dependencies, concern management, and impact of change at requirements level.

6 Conclusions and Future Work

The proposed multidimensional concern slicing approach provides a mechanism to understand the dependencies among the concerns. For each concern slice the intricate forward, backward, and parallel dependencies are analysed and semi-formal dependency equations are derived. The equations are visually represented by dependency graphs, which enable us to independently analyse the impact of change on each concern slice and level of change propagation. The early determination of change impact and reasoning regarding dependencies also help to avoid/ reduce the undesirable ripple effect propagation. Our future work will involve determining the change impact on concern slice deletion or insertion at requirements level and investigate the intensity and level of probable restructuring required at the architectural level by insertion or deletion of concern slices at the requirements level. We will also include generation of dependency graphs based on the types of

relationships among the concern slices and their sub categories. In order to calculate the transitive impact of a change, for future work we will explore the role and influence of transitive semantic dependencies.

Acknowledgements. This work is supported by European Commission grant IST-2-004349: European Network of Excellence on Aspect-Oriented Software Development (AOSD-Europe), 2004-2008.

References

1. M. Weiser, "Program slicing," *IEEE Trans. Softw. Eng.*, vol. 10, pp. 352--357, July 1984.
2. A. Rashid, A. Moreira, and J. Araújo, "Modularisation and Composition of Aspect Requirements," presented at 2nd International Conference on Aspect-Oriented Software Development, Boston, MA, March 2003.
3. P. Tarr, H. Ossher, W. Harrison, and S. M. S. Jr., "N Degrees of Separation: Multi-Dimensional Separation of Concerns," presented at International Conference on Software Engineering, Los Angeles, CA, USA, 1999.
4. A. Moreira, A. Rashid, and J. Araújo, "Multidimensional Separation of Concerns in Requirements Engineering," presented at 13th IEEE Requirements Engineering Conference, Paris, France, Aug 29 - Sept 2, 2005.
5. R. Clark and A. Moreira, "Constructing Formal Specifications from Informal Requirements," presented at 8th International Workshop on Software Technology and Engineering Practice (STEP 1997), 1997.
6. A. Moreira and J. Araújo, "Handling Unanticipated Requirements Change with Aspects," presented at Sixteenth International Conference on Software Engineering Knowledge Engineering (SEKE'2004), Banff, Alberta, Canada, June 20-24, 2004.
7. M. L. Lee, "Change Impact Analysis of Object Oriented Software," Department of Information and Software Engineering, George Mason University, Fairfax, Virginia, USA Technical Report ISE-TR-99-06, 1998.
8. K. Gallagher and J. Lyle, "Using program Slicing in Software Maintenance," *IEEE Transactions on Software Engineering*, vol. 17, pp. 751-761, 1991.
9. M. P. Robillard and G. C. Murphy, "Concern Graphs: Finding and Describing Concerns Using Structural Program Dependencies," presented at International Conference on Software Engineering (ICSE2002), Orlando, Florida, USA, May 19-25, 2002.