CCKDM – A Concern Mining Tool for Assisting in the Architecture-Driven Modernization Process.

Conference Paper · September 2012

CITATIONS
0

READS
20

4 authors, including:

Daniel Gustavo San Martín Santibáñez
Universidade Federal de São Carlos
8 PUBLICATIONS **17** CITATIONS

SEE PROFILE

Rafael Durelli
University of São Paulo
26 PUBLICATIONS **37** CITATIONS

SEE PROFILE

Valter Vieira de Camargo
Universidade Federal de São Carlos
48 PUBLICATIONS **88** CITATIONS

SEE PROFILE

CCKDM - A Concern Mining Tool for Assisting in the Architecture-Driven Modernization Process

Daniel S.M. Santibáñez[1], Rafael Durelli[2], Bruno Marinho[1], Valter V. de Camargo[1]

[1]DC – UFSCar, Caixa Postal 676 – 13.565-905 – São Carlos – SP – Brazil

{daniel.santibanez,bruno.santos,valter}@dc.ufscar.br

[2]ICMC – USP, Av. Trabalhador São Carlense, 400, São Carlos – SP – Brazil

rsdurelli@icmc.usp.br

Abstract. *The presence of crosscutting concerns in legacy systems makes the maintenance an error-prone and time-consuming activity. Architecture-Driven Modernization (ADM) is a good alternative for modernizing systems in a model-driven way by means of Knowledge Discovery Metamodel (KDM). However, the current proposal and tools did not take into account mining of crosscutting concerns. We present CCKDM, a tool for identifying crosscutting concerns which is a combination of a concern library and a modified clustering algorithm. The input is a KDM model and the output is the same KDM with the identified concerns annotated. The last one would enable to build and apply concern-based refactorings in order to have target KDM with the concerns better modularized.*

1. Introduction

Software systems are considered legacy when their maintenance costs are raised to undesirable levels but they still provide support for many organizational processes. These systems can not be just discarded because they incorporate a lot of embodied knowledge due to years of maintenance. As these systems still provide significant business value, they must then be modernized/re-engineered so that their maintenance costs can be manageable and they can keep on assisting in the regular daily activities [Harrison and Walton 2002].

In this context, OMG (Object Management Group) has employed a lot of effort to define standards in the modernization process, creating the concept of ADM (Architecture-Driven Modernization). ADM follows the MDA (Model-Driven Architecture) [Ulrich and Newcomb 2010] guidelines and comprises two major steps. Firstly a reverse engineering is performed starting from the source code and a model instance is generated. After that, successive refinements (transformations) are applied to this model up to reach the desired abstraction level. Upon this model, several refactorings, optimizations and modifications can be performed in order to solve problems found in the legacy system. Secondly a forward engineering is carried out and the source code of the modernized target system is generated. The most important artifact provided by ADM is the KDM metamodel, which is a multipurpose standard metamodel that represents all aspects of the existing IT (Information Technology) architectures. KDM is a ISO/IEC international standard (ISO/IEC 19506:2012 (ADM/KDM 1.3)).

Mining of crosscutting concerns (or Aspect Mining) is another important research field that has be exploited in the last years [Bernardi and Di Lucca 2009]. The main

purpose is to be able to automatically locate existing crosscutting concerns in a system [Durelli et al. 2013]. They are indispensable for modernization processes because most of the legacy systems suffer from the existence of a lot of crosscutting concerns spread over their architecture.

Although ADM/KDM have been created to support modernization of legacy systems, until this moment we did not find any support to facilitate the mining of crosscutting concerns using the KDM metamodel. In this paper we present a mining technique for crosscutting concerns implemented by means of a *Eclipse*^{TM} plugin called **CCKDM**. It is based on a concern library and a modified K-means clustering algorithm. The input of the tool is a KDM file and the output is the same KDM file with annotated concerns. To evaluate our technique, we used two well known systems; HealthWatcher v10 and PetStore v1.3.2 reaching precision and recall values above to 90%.

2. Tool architecture

The tool was implemented as an eclipse plugin under the Eclipse Project. In Figure 1 we present the two layered architecture of the tool.

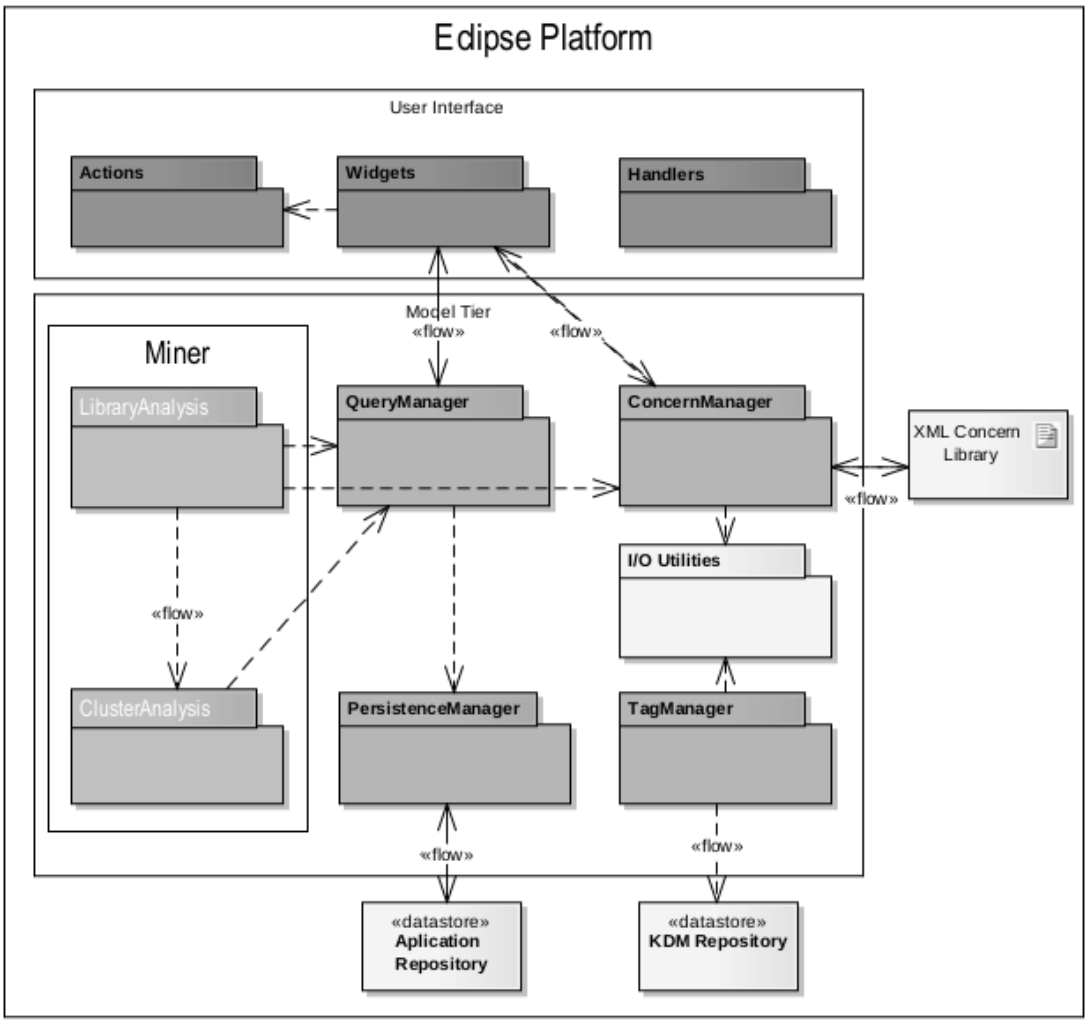

Figure 1. Logical architecture view of CCKDM

The first layer is the user interface which is composed by three packages. The package *Widgets* implements the look and feel of the tool. The package *Handlers* implements the functionality to load a KDM file and the package *Actions* implements the pop-up menu to start with the mining process.

The second layer is the model which is composed by seven packages. The package *QueryManager* handles OCL, SQL and JMQ (Java Model Query) queries. The package *ConcernManager* implements the concern library to add, delete and update concern definitions. The package *LibraryAnalysis* implements the concern mining by means the concern library. The package *ClusterAnalysis* implements cluster algorithm. The package *PersistenceManager* implements database connections. The package *TagManager* implements the annotation concern module and the package *I/O Utilities* handles I/O files.

3. Usage process and main features

In Figure 2 we depict the overall process of our technique. It is divided in three steps indicated by a capital letter into a square. The A and B steps are executed by default. Moreover, users have the possibility to activate/deactivate B and C.

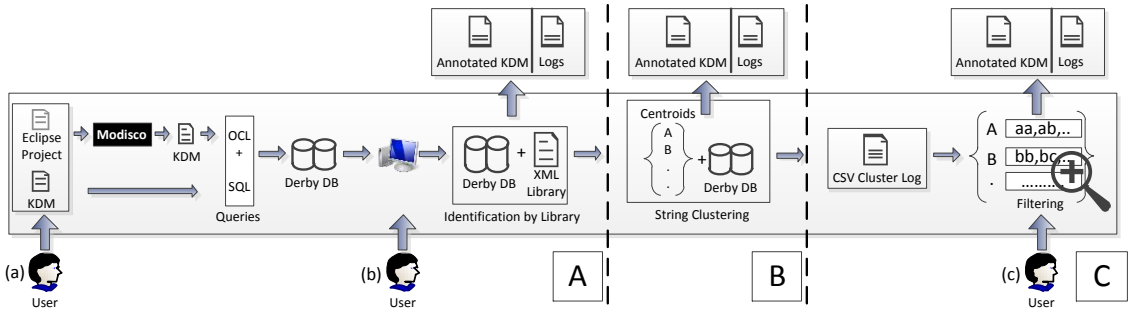

Figure 2. CCKDM process

The user in (a) starts the process by choosing an eclipse project which contains the source code or by choosing a KDM file. If the user starts the process by choosing an eclipse project, the Modisco plugin will discover and create the associate KDM model. Figure 3 shows the two ways of how to start the process.

(a) Discover KDM model (b) Load a KDM file

Figure 3. The two ways to start the process

As long as the user in (a) has triggered the process, the collect data activity begins. The tool performs OCL queries and Java Model Queries (JMQ) over the KDM model to get some code structures and store them into a database (Derby DB). When this activity is over, the main window is shown to the user. In Figure 4 we depicted the main window and for explanation purpose, we have identified 3 main regions.

Into the main window region 3, the user in (b) can select some parameters for the clustering algorithm. The first parameter provided by the tool is to activate/deactivate the clustering capability. The second parameter is to filter getter and setter methods in a way that the names of these methods will not be taken into account by the clustering algorithm. The third parameter enable the controlled annotation. That means users can choose which of the identified method and property must be annotated into the KDM file. Finally, to indicate the level of similarity between the centroids and method/property names it is used the *Levenshtein* distance [Levenshtein 1966]. If the *Levenshtein* value is closer to 1.0 then the compared strings are more similar, on the other hand if it is more closer to 0.0 then the compared strings are less coincident.

The user in (b) also must select at least one concern of the region 2 in the main window to proceed with the concern mining process. Region 1 of the main window presents some information of the loaded project, such as the number of classes, number of interfaces, number of methods and properties. These information can provide an idea of the size of the project to be analyzed. Our tool also provides information about the fan-in value of methods, where high value may indicate the presence of crosscutting concerns.

Once the user has selected the concerns to be identified, it can start the process of concern mining by triggering the button *Run CCKDM*. The step A of the process finalizes with the identification of concern seeds by using Derby DB along with the concern library (a XML file generated by our tool). Specifically, this activity consists in identifying and getting all the method names and property names which implement a given concern, matching entries of the concern library with APIs used by the application. The retrieved data is annotated into the KDM model by using BaseX (an XML database) where XQuery queries are performed. In Figure 5 we present an annotated KDM example depicting a small area of a KDM file where the *StorableUnit* attribute (a property into the source code) named "*updateHealthUnit*" is annotated with "Persistence".

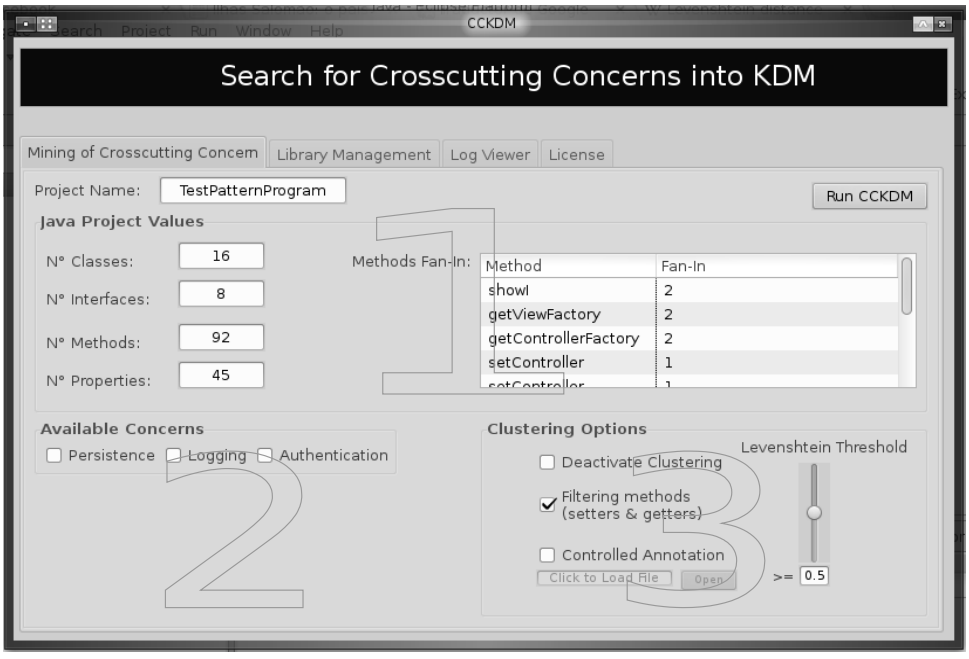

Figure 4. Main window

```
<codeElement concern="Persistence" xsi:type="code:MethodUnit" name="updateHealthUnit" type=
    <attribute tag="export" value="public"/>
    <source language="java">
       <region file="/0/@model.2/@inventoryElement.1" language="java"/>
    </source>
</codeElement>
```

Figure 5. KDM annotated

The step B corresponds to the clustering step of the process and it is performed by default but users may deactivate it. After identifying concerns by means of our concern library, our tool complements the search of concerns with a modified K-Means string clustering algorithm using the *Levenshtein* distance. All the strings identified by the concern library are the initial centroids of our cluster algorithm. Then, property names and method names that were not identified by the concern library are clustered in some centroid depending on the similarity of the strings. The concern seeds are annotated into the KDM model as we described above. Finally, the step C of the process is an alternative activity. If the user performed the step B one of the created logs is the CSV Cluster log. This log can be annotated by users to have better control on the identification process. They can indicate if the identified string name belongs or not to a concern. For that, users mark with a X letter into a CSV file as is depicted in Figure 6. After that, an annotated KDM file and logs are generated again.

** Persistence **										
close	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	
						|code|	|code|	|code|	|code|	X
** Logging **										
User	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	
logger	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	

Figure 6. CSV file to control annotated concerns into KDM file

Others features of our tool are related to the concern library management and the logs viewer. Due to space limitations, we do not include the figures showing the screens. All the files are generated inside of the analyzed project folder into the eclipse workspace.

4. Related Works

Lengyel [Lengyel et al. 2009] proposes a semi-automatic approach to identify crosscutting constraints in metamodel-based model transformations which uses a stereotyped UML model as input. Zhang [Zhang et al. 2008] proposed an aspect mining technique to recommend aspect candidates in the form of method clusters. It uses a lexical based clustering approach to identify method clusters and rank the clusters using a new ranking metric called cluster fan-in.

The main contribution and difference of CCKDM, in comparison with other existing concern mining tools [Durelli et al. 2013] is threefold; Firstly, CCKDM uses a KDM model as source of input which is an OMG standard and produce the same KDM model with annotated concerns. Secondly, it is highly customizable in order to add, edit or remove concern definitions. Thirdly, CCKDM is implemented using two combined techniques: a concern library and a modified string clustering K-means algorithm.

5. Conclusions and Future Works

In this paper we presented a new mining tool for crosscutting concerns called CCKDM which use a KDM model as input and the result is the same KDM model with annotated concerns. It is important to note this is the first work in concern mining area that use a standardized model in the context of ADM to perform search of concerns and we believe that ADM standards will be widely used in a near future because is an OMG initiative and it has the support of several important IT organizations. CCKDM has an Apache License (Version 2.0) that fits both open source and commercial software. The tool can be downloaded from the following site, `http://sourceforge.net/projects/cckdm/`. In the future we plan to improve our tool by transforming it in a framework for concern mining techniques. Thus, users could choose several mining techniques for crosscutting concerns combining static and dynamic approaches. We also are interested in integrate CCKDM with a visualization tool. This can enhance not only the identification of concerns, but also the visualization of them, allowing the software engineer analyses, in a graphical way, the parts of the software affected by concerns.

6. Acknowledgments

Daniel Santibáñez would like to thank the financial support provide by CAPES. Rafael Durelli would like to thank the financial support provided by FAPESP, 2012/05168-4. Valter Camargo would like to thank FAPESP, 2012/00494-0.

References

Bernardi, M. L. and Di Lucca, G. A. (2009). ConAn: A Tool for the Identification of Crosscutting Concerns in Object Oriented Systems Based on Type Hierarchy Analysis. In *Reverse Engineering, 2009. WCRE '09. 16th Working Conference on*, pages 319–320.

Durelli, R. S., Santibáñez, D. S. M., Anquetil, N., Delamaro, M. E., and de Camargo, V. V. (2013). A systematic review on mining techniques for crosscutting concerns. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 1080–1087, New York, NY, USA. ACM.

Harrison, M. S. and Walton, G. H. (2002). Identifying high maintenance legacy software. *Journal of Software Maintenance*, 14(6):429–446.

Lengyel, L., Levendovszky, T., and Angyal, L. (2009). Identification of crosscutting constraints in metamodel-based model transformations. In *EUROCON 2009, EUROCON '09. IEEE*, pages 359–364.

Levenshtein, V. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707.

Ulrich, W. M. and Newcomb, P. (2010). *Information Systems Transformation: Architecture-Driven Modernization Case Studies*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Zhang, D., Guo, Y., and Chen, X. (2008). Automated aspect recommendation through clustering-based fan-in analysis. In *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, pages 278–287, Washington, DC, USA. IEEE Computer Society.