Domain Mapping for Product-line Requirements

Kendra Schmid
Iowa State University

Follow this and additional works at: http://lib.dr.iastate.edu/cs_techreports
Part of the Software Engineering Commons

Recommended Citation
http://lib.dr.iastate.edu/cs_techreports/336

This Article is brought to you for free and open access by the Computer Science at Iowa State University Digital Repository. It has been accepted for inclusion in Computer Science Technical Reports by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Domain Mapping for Product-line Requirements

Abstract
This paper investigates the effectiveness of the domain mapping technique to enhance the requirements specification of product lines. Domain Mapping uses glossary terms and relations among software components to determine common definitions and requirements that are applicable across products.

Keywords
domain mapping product-line

Disciplines
Software Engineering

This article is available at Iowa State University Digital Repository: http://lib.dr.iastate.edu/cs_techreports/336
Abstract
This paper investigates the effectiveness of the domain mapping technique to enhance the requirements specification of product lines. Domain Mapping uses glossary terms and relations among software components to determine common definitions and requirements that are applicable across products.

Introduction
As systems become more complex and inter-reliant, there is an increased need for complete and accurate documentation [5]. There must exist a broad, general base of comprehension such that the transition from requirements documentation to implementation is carried out in the appropriate and intended manner. The ideal option is to locate the inconsistencies and missing information early in the life cycle in order to appropriately mitigate the problem before it arises.

In the cases of product lines, the general base of knowledge must assume enough depth to be beneficial for each product yet provide expression of assumptions for all components that inherit from it.

Thus, this paper shows the beneficial use of analyzing the requirements documents using the domain mapping technique for product lines.

Background
Domain mapping.
The goal of creating a domain map (DM) [1] is to define terms and relationships in the software specifications in terms of common usage and semantics. A domain map tree provides easy access to the definitions of the terms thus allowing a non-expert more complete and consistent information while attempting to create the product. The tree is essentially an expandable outline structure where at each level the terms that are used to define the parent term are explained. The idea is that a person can navigate through the tree to find the basic concepts in basic language. Figure 2 shows an excerpt from a domain map.

To thoroughly define the terms, the developers and experts in the domain must come to agreement on the definitions. This often involves an iterative process of adding specifications and restrictions on the terms until a suitable agreement is reached. This can be in the form of relationships to other words, textual descriptions, or references.

Advantages:
Domain mapping allows for a more complete understanding of the terms involved. This understanding leads to a standard starting point that all systems can use. If used properly
as a reference tool for terms, it can eliminate problems arising from assumptions made about a term. A simple example would be defining all distance measures to be in kilometers versus miles to eliminate misunderstandings regarding units of distance.

The process of creating the domain map itself assists understanding by systematically stepping through the requirements and terms. The domain map simplifies terms to common semantics that clarifies some assumptions by showing what definitions the term builds on which provides additional context for the assumption. Compiling the terms into a common location, makes inconsistencies evident. This process acts as an initial flag for a check of related inconsistencies and further explanation that may be required.

Product Line
A product line is a series of related software systems that share common base architecture. [4] What makes the applications different from each other are the variations of components and specifications for the specific type of product. These products are developed to reuse portions yet allow variations from a standard design. This allows for customization at a lower cost than rebuilding and redesigning for each new component.

For example, all automotives could be looked at as a product line. They all have four wheels, an engine, doors, steering control, brakes, etc. However, each of these components is not the same across all cars. Some cars have two doors while others have four, which makes these products different. Though they still share some standard design of how the body, seats, and wheels should be laid out.

Commonality Analysis
Commonality analysis (CA) consists of a definition of technical terms, commonalities among sections, Variabilities among sections, and parameters of variation [3]. Figure 1 shows an excerpt from a CA.

Commonalities are shared features or components that occur in all instances of the products. For example, “there exists a display console in the system” is a commonality across all driver displays.

Variabilities are the features or components that may vary within the instances of the product. For example, “the number of display consoles may vary” is a variability.

The parameters of variation describe the values that the components may take. For example, [1..3] means that there may be one, two, or three instances of the display console placed into the system. The parameters of variation also include a binding time at which the variation must be decided. Some examples of the binding time would be specification time and run time.

Case Study
The case study for this report was a requirements specifications document of a fictional driver monitor system. The document was created for a workshop at The Second Software Product Line Conference held at Carnegie Mellon University from on August
19-20th 2002. The purpose of the document was to provide a case study that participants could analyze for possible improvements.

Driver Monitor System
Driver Monitor System (DMS) is a system that is designed to be a product line of on-board monitoring of the vehicle and driver. Major variations that make this system a product line is the vehicle of implementation, for example car versus truck, and the purposes for which the system is used, for example on-board diagnosis, maintenance, and driver training to list a few.

The system consists of a central repository of control and information with subsystems to control the data gathering and implementation of use cases. The subsystems include behavior monitoring, display, vehicle dynamics measuring, engine monitoring, controls, and driver action monitoring. Some of these subsystems are further broken down as well to allow for further variation.

Developing the Technique
To compare the techniques of Commonality Analysis and Domain Mapping both techniques were applied to the DMS specifications. The primary step in DM is for an agreement of the domains. In the paper by Hanks, et al.[1] it is assumed that separation of perspectives exists purely between developer and expert. However, the complexity of product lines introduces multiple subsystems of developers that need to interact consistently with each other as well as with the experts. In this case study, the classifications of domains are the separate subsystems of the DMS such as the controls, driver action, and vehicle dynamics. This partition into domains was done because the existence of each subsystem may or may not exist in a product, most subsystems are developed separately.

Next, a domain map tree was assembled by stepping through the document and recording the information associated with terms. This collection of information takes the form of a collapsible tree that shows reference to the other terms used to define it.

In this case study, the terms chosen were entries from the current glossary of the specification. From there, relationships and best-fit explanations were constructed by propagating through all connections a term connected to other sections in the document. This was a manually intensive process since it involved searching repeated visits to a few frequent sections.

The next portion of the DM technique is to have an expert analyze the DM for incorrect definitions and missing terms. For this case, no such expert was available. Instead, the assumption was made that any ambiguous terms were sources of confusions that could have resulted in errors or questions to experts.

When sufficient information for a portion of the system does not exist, the technique of creating the domain map relies on outside specialties to input information that might not
exist in any other section of the documentation. The tree creates a common repository for the expert knowledge. [2] This includes information on motivations to prevent miscommunications associated with the possible adjustment of terms in one section without jeopardizing another area.

During the creation of the DM, it is possible to discover and document local inconsistencies of terms and relationships. This occurs by noticing an anomaly during the addition of information to an entry in the DM. After the DM is created, it would be beneficial to step through the terms in search of global inconsistencies that contradict and affect the different domains. In this case, this was done at the same time as local inconsistencies due to the creation of a single inclusive DM. This is slightly problematic that the person who updates the document must be vigilant to observe these errors, or a separate walk-through may need to occur.

By using this technique, the CA may be done much quicker since much of the information is redundant with that of the DM. For instance, a commonality of “Display can consist of alternate (Red/Green zones) and multi-function displays.” also appears in the entry of Display in the DM. In addition, the DM immediately picks up the variation of “The number of Display subsystems may vary” instead of needing to search through both parts of the CA. However, the parameters of variation do not appear in the DM in the concise representation.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
<th>Domain</th>
<th>Binding Time</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2: Disp, V2</td>
<td>Number of Display subsystems</td>
<td>[1...∞)</td>
<td>Specification</td>
<td>1</td>
</tr>
<tr>
<td>P9:RGZones,V9</td>
<td>Number of Red/Green zone display</td>
<td>[0...∞)</td>
<td>Specification</td>
<td>1</td>
</tr>
<tr>
<td>P10:MultDis,V10</td>
<td>Number of Multi-function displays</td>
<td>[0...∞)</td>
<td>Specification</td>
<td>1</td>
</tr>
<tr>
<td>P15:DispComplex,V16</td>
<td>Complexity of display</td>
<td>[Red/green zones… full data analysis]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. Class: Display
   a. Alternate (Red/Green Zones)
      i. Class: Red/Green Zones
      ii. Use case: View Red/Green Zones
         1. engine performance
            a. Use case: Monitor engine performance
               i. RPM
               ii. temperatures
               iii. Exhaust levels
               iv. Monitor actor
            v. Engine Monitoring diagram
   1. Interface Device: Engine Monitor
      a. RPM
      b. Fuel consumption
         i. Use case: Monitor Fuel consumption
         ii. distance
         iii. Driving type
         iv. Monitor actor
         v. Engine Monitoring diagram
   c. DMS

2. Driver actor
3. Monitor actor
4. Display viewing diagram

b. Multi function display
   i. Class: Multi-function Display
      1. Use case: view multi-functional display
         a. Monitor actor
         b. driver actor
         c. Display viewing diagram
   ii. Use case: view multi-functional display
      1. Monitor actor
      2. driver actor
      3. Display viewing diagram

Figure 2: Domain Map of Display

Results

Local Inconsistencies
Local inconsistencies are discrepancies among concepts within a domain itself. This includes relationships that should act in a particular way, but are not reflected in the documents. For instance, if in the term ‘max speed’ is defined as ‘80 mph’ then later in the same domain it is redefined at ‘60 mph’, then there would be a local inconsistency.

In the case study of the DMS specifications, on page 24: “Behavior Monitoring is made up of Driver Monitor System” is stated as one of the descriptions. However, on page 26: “Each Driver Monitoring System is made up of Behavior Monitoring.” This leads to a circular requirement. In this case, additional explanation would be requested from the expert.
**Global Inconsistencies**

Global inconsistencies are the discrepancies that occur between domains. For example, if one domain assumed all values of distances to be measured in meters where another domain assumed all values to be in yards, then the concept of ‘distance’ would be globally inconsistent. These are especially problematic for product lines where reused components need to share definitions to have complete and compatible structures. None were found in this case study since the domains were tightly coupled. In addition, no corollary documents were referenced since this was created as a stand-alone example.

**Ambiguities**

Ambiguities are the general terms or relationships that are not defined sufficiently. This often includes being too vague about the function, units, or relationships of the components. For example, if a function is designed to ‘get rid of all unused components’. It is ambiguous as to what ‘get rid of’ should do whether it is delete completely, store in a back up, or not consider in the equation.

From the example, the vehicle dynamics uses the term “2-axis acceleration (transversal, longitudinal),” but there is no further description or explanation of it. It is ambiguous in the components needed to produce or use these values. However, it is expected that this could be rectified through expert interaction.

**Orphans terms**

Orphan terms are requirements or terms that are defined, but not used. For example, if the glossary defined ‘fast’ as being ’80 mph’ but then never used the term ‘fast’ in the document. This is often an indication of some intention that may not be complete. In reference to the ‘fast example’ it would indicate that some component should have been using this term but this component is missing.

As an example, “heart rate monitoring” is defined as a subset of behavior monitoring and has a description. However, the term is not connected to any further discussion or use of behavior monitoring.

**Multi-domain and cross-cutting features**

The architecture behind the program deals with the multi-domain and cross-cutting features. In the example used, there was a centralized control and database. This virtually eliminated cross-cutting features by making each section its own black box and not relying on any other domain. This would allow for more variability of the subsystems of the program since if one sub system were removed it would not directly affect any other portion. Hence, any multi-domain features are dealt with in the central repository to direct and deal with Variabilities of systems.

**Coupling and correlation**

After a domain map is created, it allows for easy access to information regarding coupling and correlation. Coupling is when items are dependent upon each other. Domain mapping shows coupling by the relationships that terms and sections have with each other, such as ‘kinetic energy’ being defined in terms of weight and velocity. In
addition, the use case of Monitor engine performance relies on fuel consumption monitor, so these are closely coupled.

Correlation is how internally coupled an item is. For example, the item has only what it needs and needs everything it has. Correlation within the specifications can be viewed from the tree by referring to how often the terms reflect back on themselves. This is due to the nature of mapping because it expands the sub trees to view the hierarchy of elements that reference to the original term, which indicates the terms are frequently associated. However, these loops provide no increase in clarification of definition. [1]

**Depth of information**

A useful feature of domain mapping is that it illustrates the depth of unique information available for any given terms. First, by looking at the sheer quantity of information and explanation present, it can be determined how much is known about the term. Second, it is possible to see the depth of the term, i.e., how many relationships it builds on, and can be used to learn more about the term. This depth also can be used as a metric to see if one section is more well-defined because it has the relationships and information.

**Continued ambiguities**

If the term was only defined in relationships in the first place, the first round of domain mapping will not provide any additional information, especially if all of the other terms also only use relationships to define their use. This definition would be incomplete since it does not provide any greater understanding than just looking at a chart of lines and their connections. An example of this occurs in the Monitor Actor description, where most of its terms refer to a use case which then refers back to monitor actor.

**Discussion of Results**

The use of the domain mapping technique on product line requirements documents yields additional insights into the requirements over a CA alone. The extra time spent walking through the requirements documentation and creating a reference document identifies many inconsistencies and possible areas of miscommunication. By doing this, it opens a channel for communication between the various domain groups. Once completed, the DM can act as a source of authority to standardize components and definitions across the variations that occur within product lines. This can then replace the need to track down members of differing groups for every incremental need of explanation.

The time invested in DM reduces the time invested in CA since the grouping of information in the domain map would aid the creation of the CA. However, neither technique can replace the other since the information gathered and displayed differs.

The amount of effort in a DM is greater due to the formatting that occurs. However, a tool proposed in the Hanks paper would lessen this strain. In addition, the DM does require more time to contact the experts or other groups.
Further area of study
To further develop this technique and weight the benefits, it would be of interest to apply DM to a case study in which the structure of the component interaction is not based upon a central repository, thus allowing research into finding cross-cutting inconsistencies. Additionally, it would be of interest to develop the technique on a case study involving more detailed information that originates from multiple experts and done in an iterative manner which would provide cross-expert information to appear. This detailed information would provide a challenge to the use of DM due to the possibly large volume of specific information and whether that would overflow the system of circular references.


*This research was funded by National Science Foundation Research Experience for Undergraduates Grants #0311876 and #0311877.

Acknowledgments:
This was collaborative work with Robyn Lutz.