Feature Modeling and Automated Analysis for a Digital Television Product Family

Gülseren Fedakar Gönül, Halit Oğuztüzün, and Ahmet Serkan Karataş

***Abstract*—This paper presents a feature modeling endeavour for a television set product family by a major consumer electronics company. This work consists of three stages. First, a feature model is constructed, based on the analysis of the product family requirements. The constructed model is supplemented with a feature glossary. FeatureIDE is used as the model editor. Feature attributes, not supported by FeatureIDE, are represented in the basic feature model by using additional features. Second, the feature model in XML is converted into the schema of the analysis tool, using a custom parser developed for this purpose. Third, the model is analyzed by well-known analysis operations. FAMA is used as the analysis tool. Performance results are obtained. Finally, lessons learned from the whole effort are discussed.**

***Index Terms*—Feature modeling, extended feature model, software product lines, commonality, variability, variability management.**

I. Introduction

In consumer electronics industry the focus is shifting from single products to product families, thus, from project organization to product line organization. Thanks to the Software Product Line Engineering (SPLE), it gets easier to configure products for various customer needs, thus, the time for producing new products reduces. In the context of software product line engineering, feature models are used for modeling variability and commonality in product families. Feature modeling was proposed as a part of the Feature Oriented Domain Analysis (FODA) [1]. FODA defines important and distinctive system characteristics, which influence user experience, as features. Variability and commonality in a product family can be expressed and managed using feature models. In [2] it is stated that inspecting commonality and variability in a systematic way during the domain analysis supports product family identification.

A digital TV [3]-[5] can be described as the point where the computer and the television meet. Similar to computers, TVs that possess digital TV characteristics have an operating system, internet access, online services, software applications, etc. In digital TVs, plenty of operations can be realized by adding new features to them. These features are in a wide variety such as picture transfer via USB/DLNA from mobile phone, tablet or PC, saving a broadcast by television as a video without needing any external device, video-conference systems, connecting to the internet via TV, using social media applications, etc. Digital TV product line serves as a challenging example for variability and commonality management in the framework of SPLs as there is an increasing demand in the sector due to changing customer needs and expanding set of features that the products can have.

In this study, variability and commonality in an industrial embedded software product family is modeled using feature models. "Smart TV" product line of a major consumer electronics company in Turkey, which is a digital TV and will be referred here as the NewGenTV product line, is used as the case study. First, we constructed the feature model for the NewGenTV product line. During this process we used FeatureIDE [6] as the modeling editor. As FeatureIDE does not support feature attributes but some of our features have them, we adopted a method, which will be discussed in Section III, to define feature attributes as features. Next, we performed a translation which takes feature model in XML format as input and converts it to the AFM format of FAMA [7], [8]. Finally we used FAMA to perform automated analyses on the model.

Manuscript received December 10, 2013; revised April 8, 2014.

The authors are with the Department of Computer Engineering, Middle East Technical University, Ankara, 06531, Turkey (e-mail: gulseren.fedakar@gmail.com, oguztuzn@ceng.metu.edu.tr, karatas@ceng.metu.edu.tr).

II. Background

FeatureIDE is an open-source framework for feature oriented software development (FOSD) [6]. It is built on Eclipse and can work smoothly with large models. It has a GUI view that enables model editing via a graphical interface. It can generate an XML-based representation of the edited model. However, its present version does not support feature attributes.

FAMA is an automated analyses framework for feature models. FAMA integrates a number of commonly used reasoners such as Choco, Sat4j, JavaBDD, and JaCoP, where it selects the most efficient solver automatically in run time for the best performance considering the operation in use. It can operate on both the basic and the extended feature models. In FAMA, domains of the attributes consist of integers. Moreover, attributes can be used in constraints.

III. Feature Modeling

We started our study with the domain analysis phase. In this phase, we adopted the strategy reported in [9] and initially completed the domain characterization and project planning tasks. Then, we performed domain analysis steps such as data collection, data analysis, classification, and evaluation of the model iteratively. Thus, we revised our model repeatedly. During these iterations, we identified the features and their attributes, feature grouping characteristics,

DOI: 10.7763/IJCEE.2014.V6.853

common and variable features, relationships and dependencies between the features, and the constraints in the family. To overcome the problems we encountered, we performed an extensive literature survey on the related fields, conducted interviews with the domain experts from the company, and received continuous feedback from the experts, who constantly reviewed the evolving feature model. For the data collection step, we started by collecting information from the domain experts, product documentation, and publications on digital TVs [3]-[5]. This data were mainly a bunch of vague information for the people other than domain experts before we chose appropriate ones and extracted them as features. Next, for the analysis step, we analyzed the collected information in order to detect the reusable features and identify some of the similarities and differences among them. We also started to build a Feature Glossary, which has been repeatedly revised along with the model as the domain analysis continued, that explains terms, concepts, and standards about the domain that appear in the feature model such as DVB-S, HDMI, TV Tuner, PIP, PAP, etc. Then, in the classification step, we abstracted the features according to their descriptions and relevance to common concepts, and created groups that consist of similar features. Finally, after eliminating the errors, contradictions, and gaps, we obtained a primary model that presents 23 conceptual groups when all characteristics are considered. These conceptual groups were presented as children of root feature as seen in the Fig. 1. Moreover, the model also included the primary dependencies in the form of constraints.

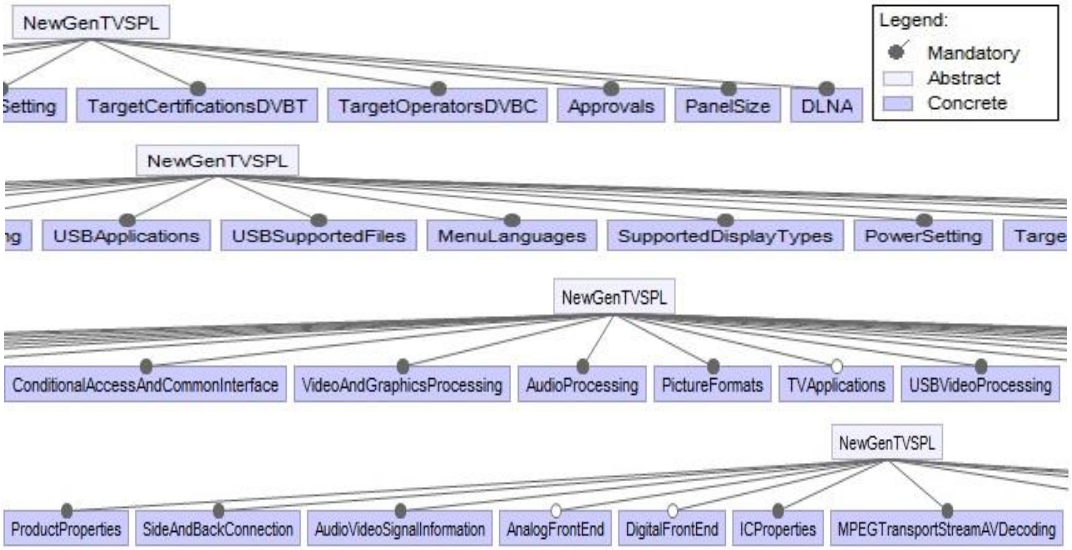

Fig. 1. Conceptual groups of feature model.

In the following subsections we discuss some of the key tasks in the domain analysis phase, and finally present the important characteristics of the resulting model.

A. *Feature Identification*

Identification of features involves abstracting domain knowledge obtained from domain analysis. In general, features can be classified into four categories: capability features, operating environment features, domain technology features and implementation features [10].

Capability features typically have user visible properties [10]. We identified 345 capability features in NewGenTV. This category includes features such as ChannelBandwithForAnalog, which is a non-functional characteristic of the system, ManualSearch, which is a distinct service provided for user and listed in the user manual, and AFT, which is an internal function to provide a service.

Operating environment features are the features which are related to the environment (hardware and software) in which the system is used and are related to communication of this system with external systems [10]. We identified 14 operating environment features in NewGenTV. For example, in our model CPUPower and RAMPower are the operating environment features. Also LineoutOutput feature is operating environment features as it is related to the port that speakers connected to.

Domain technology features are the features which are specifically related to the domain and may not be meaningful in other domains [10]. They indicate the techniques of implementing operations or services. We identified 221 domain technology features in NewGenTV. This category includes features such as DVB_T, which is a Digital Video Broadcasting standard for television, and NTSC, which is an analogue television color encoding system standard.

Implementation features are generic functions or technologies which can also be used in other domains [10].

They are used to implement other features. We identified 17 implementation features in NewGenTV. This category includes features such as LNBpowerAndPolarization_DVBS2, which is used to implement a satellite TV, and DiseqC12Support, which is a communication protocol for use between a satellite receiver and a device such as a multi-dish switch or a small dish antenna rotor.

B. *Feature Model Organization*

After the feature identification step a hierarchical model is created by classifying and structuring the features using their relationships. First, we have identified 23 conceptual groups, groups of features that are related to each other, in

the model. Then, we added each feature as a child to the feature that represents the conceptual group it belongs to.

The elements of this feature model have basic decompositional relationships. These are mandatory-relationships, optional-relationships, or-relationships, alternative-relationships as seen in the Fig. 2. Also cross-tree constraints, which allow for the specification of further dependencies among hierarchically unrelated features, are added to the model. For example, as seen in the Fig. 3, the constraint DigitalProgramStorageT_C implies (DVB_T or DVB_C or DVB_T2). As DigitalProgramStorageT_C is only applicable to terrestrial or cable broadcasts, it implies that DVB_T or DVB_C or DVB_T2 is selected.

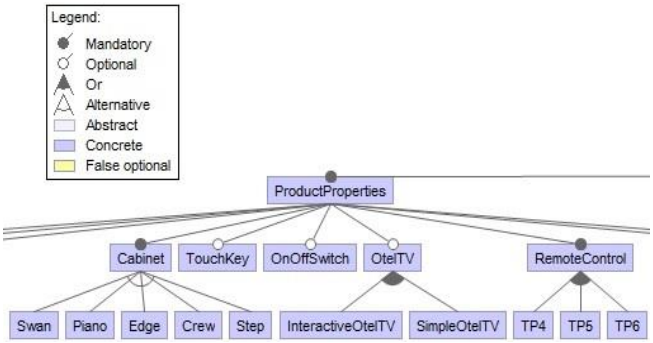

Fig. 2. Feature model relationships.

DVB_S2 ⇒ DVB_S

_3DCombFilter ⇒ NTSC ∨ PAL

DVB_T2 ⇒ DVB_T

mkv ⇔ divx

T ⇒ DVB_T

DVB_S ⇔ SiliconTuner

DVB_C ∧ DVB_T ⇔ HybridCanTuner

DVB_C ⇔ InputFrequencyDVBC ∧ InputLevelDVBC ∧ ModulationDVBC ∧ SymbolRateDVBC ∧ Full_Quick_NITSearch_DVBC

Full_Quick_NITSearch_DVBC ⇒ DVB_C

DigitalProgramStorageS2 ⇒ DVB_S2

DigitalProgramStorageT_C ⇒ DVB_T ∨ DVB_C ∨ DVB_T2

MHEG_HD ⇒ DVB_T2

Fig. 3. Constraints.

C. *Defining Feature Attributes*

FeatureIDE version 2.6.2 [6], which is the version we used in this study, has no representation for features with attributes. However, in our model there exist some features with attributes. Therefore, we adopted a method to represent feature attributes in FeatureIDE. Using this method, we constructed a basic feature model to mimic the extended model we have.

For each feature that has attribute(s), a new feature that denotes a variation point is added as a child, connected using the mandatory relation to its father, the feature with attribute(s). We use the prefix "vp" in the names of such features to indicate that they represent a variation point. In general, the naming convention we use for the variation point features is as follows: "vp"+ parent feature name + "_in"+ unit type. Next, we added a new feature for each possible value that an attribute can take, and connected these new features (i.e. variants) to the variation point feature. When there is only one possible value we used the mandatory relation for the connection. On the other hand, when there are two or more values, we used or relation since a product can support any non-empty subset of these values. The naming convention we used for the features representing possible attribute values are as follows: parent feature name + "__" + value. For example, as presented in Fig. 4, the feature ArealInputImpedance has an attribute that can take the value 75 Ohm and the feature ChannelBandwidthForAnalog has an attribute that can take two possible values, namely 7 MHz and 8 MHz.

When the possible value an attribute can take is a range, we applied a slight modification to our naming convention, and inserted the string "to" between the end minimum and maximum values of the range. For instance, if the value an attribute can take is the range 47-77, then we used "47to77" as the possible value as seen in the Fig. 5.

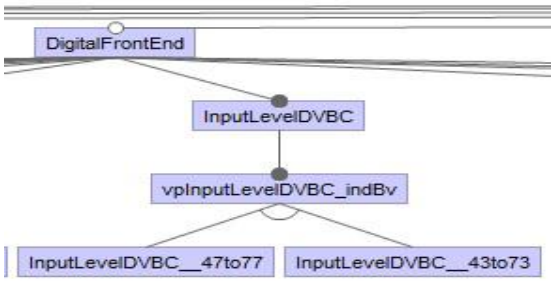

Fig. 5. Attributed model for ranged values.

At the end of modeling process, the feature model we obtained contained a single root feature named NewGenTVSPL (level-0) and 23 main feature groups (level-1). After adding all the features to the model, we obtained a model that has 621 features in total, including the root feature. The feature diagram that represents this model has 427 internal nodes (i.e. 427 features are further decomposed into other features) and 194 leaves. The longest path from the root feature to a leaf feature consists of 4 decomposition relations. The average number of children for parent features is 2.2. However, there are some extreme cases. For example, the feature ICProperties has 42 child features. There are 504 decomposition relationships in the model, where 238 of them are mandatory-relationships, 234 of them are optional-relationships, 16 of them are or-relationships, and 16 of them are alternative-relationships.

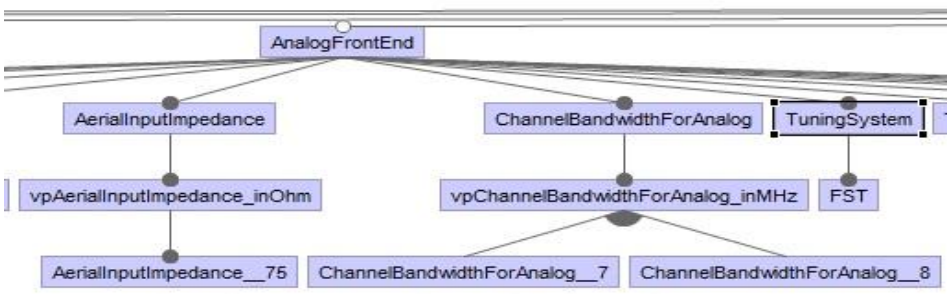

Fig. 4. Attributed model for AnalogFrontEnd.

There are 36 cross-tree constraints in the model. FeatureIDE allows defining constraints with the logical operators such as implies, iff, and, or, not. For example, in the feature model "DVB_S implies QPSK" was defined as constraint because in DVB_S the data is modulated on the carriers with QPSK. Although we cannot use excludes operator in FeatureIDE, using iff and not operators achieves the same effect. Our most complex cross-tree constraint has 9 logical connectives: DVB_S2 iff (TransportStreamDVBS2 and ProfileLevelDVBS2 and AudioDecodingDVBS2 and AudioModeDVBS2 and SamplingFrequencyDVBS2 and DataRatesDVBS2 and DigitalFrontEnd_DVBS2 and Demodulation_DVBS2 and SignalLevel_DVBS2). The Feature Glossary constructed in the domain analysis stage was utilized in this process as it provides information about the features.

IV. Transformation

Having constructed the model with FeatureIDE, we now have an XML based feature model to be analyzed in FAMA tool. Therefore, at this point we need a translation from the feature model in XML to the FAMA input format AFM. A parser was developed for this purpose. DOM which is the Standard Java API was used. The javax.xml.parsers package which is a JAXP package was used to obtain a DOM parser and to parse the file into a DOM Document object while org.w3c.dom package and its subpackages were use for traversing the document.

The parser takes the feature model which is in XML format as an input and transforms it to the AFM format. The whole XML document was parsed as a Document Model. After that, necessary information was subtracted and transformed to the corresponding AFM components. Below we presented some diagrams which were drawn in a drawing tool [11] for a clear understanding of the FeatureIDE's format and its corresponding component in AFM format derived using parser. In Fig. 6, general representation of transformation can be seen on an example.

Parser also identifies feature attributes considering naming convention mentioned before. According to our naming convention, the name of this child features start with "vp". Actually the reason of this was to indicate that its child is attributed feature. The value of the attribute was extracted taking the number after the double after the double underscore character. The attribute value was written after the line %Attributes. According to FAMA's format, a feature's attributes are defined with their type, domain default value and null value which, is the value when the feature is not selected.

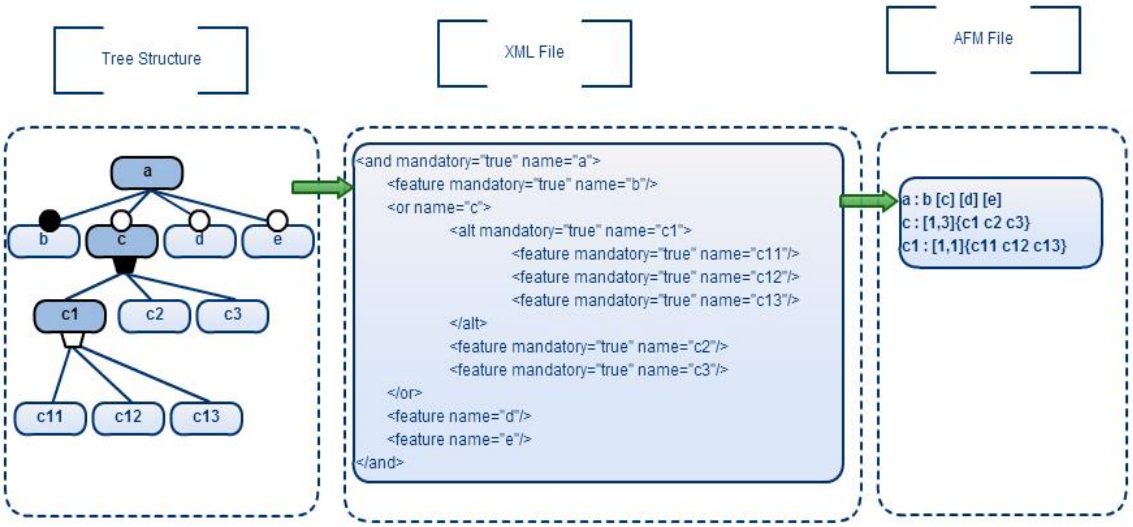

Fig. 6. Mapping of complex relationships.

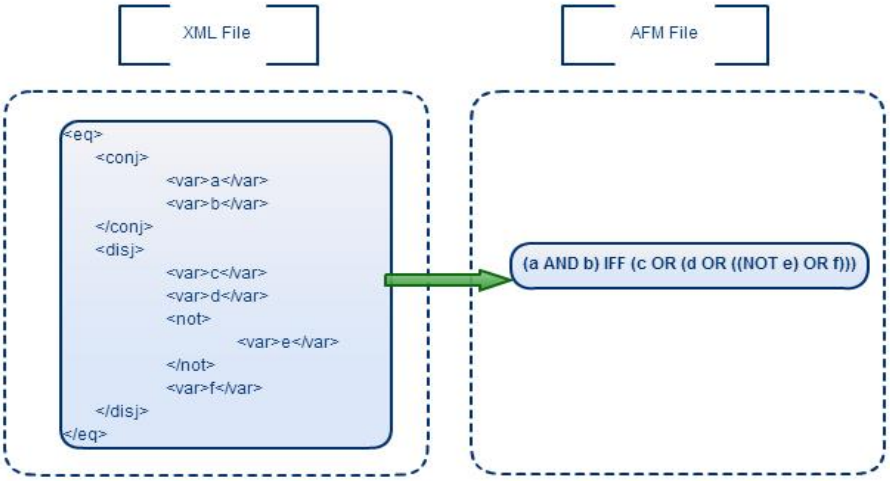

Fig. 7. Mapping of complex constraints.

V. Analyzing the Feature Model

We used FAMA standalone distribution, version 1.1.2. It is a Java library which can be easily integrated into Java projects.

FAMA operations applied in this study are Validation, Products, Number of Products, Valid Product, Invalid Product Explanation, Valid Configuration, Error Detection, and Error Explanations [12]. FAMA can use different solvers (Choco, ChocoAttributed, Sat4j, JavaBDD and JaCoP). However, presently, only Choco provides both standard and extended model support. Therefore we used Choco [8] for all performance measurements.

The run time of the operations is about a second using the full model in most cases. For the Invalid Product Explanation, it depends on the structure of the input product. When input product has too many options to repair an invalid product for a given model, it takes more time. For example, in our model, when this question gives 6 different suggestions for an input, the run time is 10 seconds. However for Products and Number of Products questions, we were encountered a performance problem. As number of features in our model is high, all possible products that can be extracted from the model can be tremendous. (See Fig. 7).

Actually it is impossible with an ordinary personal computer. The properties and the hardware configurations of the PC by which this study was done as follows: Windows 7 OS, Intel Core i7 CPU, 2.20 GHz, 8 GB RAM, 64 bit OS, 400 GB HDD. Therefore, to test Products question, a very reduced version of our feature model was used as input After that we increased feature number in the model and performed this analysis again and again. Some of the results can be seen in the Table I. When the model gets a bit more complicated, # of products increases exponentially. Another observation that can be deduced from these two analyses is that when we just increased variant feature -added a variant feature to the model-, both time spent and number of products were increased whereas when we increased core feature number, time spent was increased but number of products did not change. Moreover a variant feature can be a child with or/and/alternative-relationships. As an example, we can give out27.afm, out27v2.afm and out27v3.afm feature models. For example, Out27.afm was constructed by adding 2 variant features which have or-relationships to the Out25.afm. For this input, number of products became 24. Out27v2.afm was constructed by adding 2 variant features which has alternative-relationships to the Out25.afm. For this input, number of products became 16. Out27v3.afm was constructed by adding 2 variant (optional) features which have optional-relationships to the Out25.afm. For this input, number of products became 32. It can be deduced that adding variants which have optional-relationships increases the number of products drastically whereas adding variants which have alternative-relationships increases the number of products mildly as seen in the Table I.

In this study, for a reduced feature model with 70 features, Choco calculated all the 3491840 products as seen in the Table I. When we add more features to this model, we saw that the same analysis takes more than 6 hours. Therefore we had to stop the analysis runs with larger models. Number of Products question is, in essence, the same as the Products question. However the only difference is in the time spent as seen in the Fig. 8, because calculating all possible products takes much more time than just calculating number of those products. In the Fig. 8, the y-axis shows the time spent in minutes. Therefore analysis of number of products can be done with the models with more features. In our model which has 79 features, the run time of this analysis takes 295 minutes. However, when the variant feature number is increased at each input, time spent for run is also increased. For example, in the model having 80 features, it took more than 6 hours. So it is not practical to analyze more crowded models with this question.

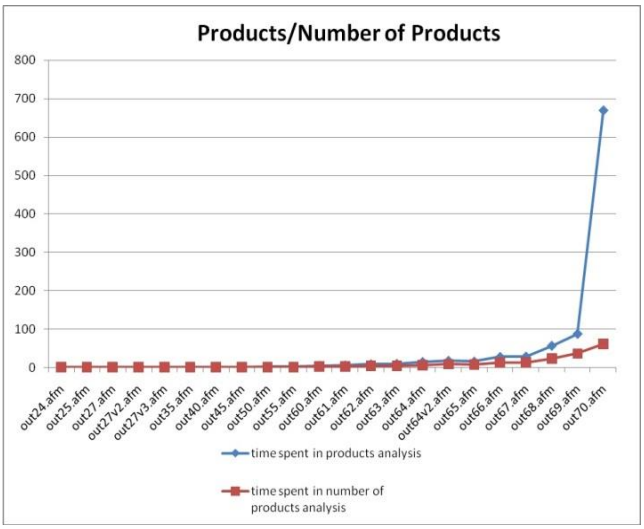

Fig. 8. Products/number of products run time analysis diagram.

TABLE I: Products Analysis

Input file	# of featu-res	# of core featu-res	# of variant featu-res	Time spent	# of products
Out24. afm	24	21	3	1 seconds	8
Out25. afm	25	22	3	1 seconds	8
Out27. afm	27	22	5	1 seconds	24
Out27v2. .afm	27	22	5	1 seconds	16
Out27v3. .afm	27	22	5	1 seconds	32
Out35. afm	35	30	5	1 seconds	24
Out40. afm	40	30	10	1 seconds	768
Out45. afm	45	35	10	1 seconds	768
Out50. afm	50	35	15	2 seconds	23808
Out55. afm	55	40	15	3 seconds	23808
Out60. afm	60	40	20	5 seconds	119040
Out65. afm	65	43	22	16 seconds	396800
Out70. afm	70	44	26	11 minutes	3491840
Out71. afm	71	44	27	More than 6 hours	

Using FAMA analysis, we can improve models or fix invalid products. For example, Invalid Product Explanation provides suggestions to fix an invalid product. Error Detection question detects erroneous cases, such as dead

feature, false mandatory, wrong cardinality and void model. Error Explanations question looks for explanations for detected errors in terms of relationships. For example, when we ran our model with Error Detection, we got false mandatory error. Running Error Explanations analysis we got the following results:

False-mandatory Feature: HDReady
Relation: HD -> FullHD HDReady
Relation: HD -> HDReady
Relation: FullHD IMPLIES HDReady

As seen in the results, HDReady is a false-mandatory feature. This analysis not only tells that HDReady is a false-mandatory feature, but also gives possible reasons of this error. The relations seen in the outputs are the explanations for the errors. They help us to detect the error reason and fix the error. The analysis gave us three relations. When we inspected first two relations, we saw that HD is the parent of two features namely FullHD, HDReady and these children have or-relationships. Therefore they are not mandatory. However, when the third relation (FullHD IMPLIES HDReady) was inspected, it can be deduced that HDReady should be selected in every possible product. Actually there are three possible product configurations related to those features. As at least one of the children of HD should be selected, for the first configuration, we chose FullHD. Because of the constraint (FullHD IMPLIES HDReady), HDReady should also be selected. For the second configuration, we chose HDReady. Lastly, for the third configuration both of these children were selected. Therefore, in all there possible configurations, HDReady was selected. So it is false mandatory, in other words, it must be in every product, although it was defined as optional.

In the light of this error explanation, some changes were made to the model. As told earlier, HDReady should be mandatory. Therefore or-relationships had been converted to mandatory-relationship and optional relationship. HDReady was made a mandatory feature and other child features were made optional.

VI. Conclusion

In this article, a feature model for a commercial TV set family has been introduced for dealing with the commonality and variability in terms of feature models and enabling automated analysis. Although our SPL domain is applicable to an extended feature model as it has attributes, in the modeling part we used basic feature model. However, in the translation part, this model was converted back to the extended feature model and analysis operations were applied on this extended feature model.

In the modeling part of the study, we used FeatureIDE. Although graphical modeling tools make the modeling process easier for non-technical people such as customers, they are not efficient and sometimes not fully functional while working with large feature diagrams and complex constraints. Besides, the capability of FeatureIDE is limited such that expressing attributed feature models is not possible. As the model gets larger, using constraint editor gets harder. Sometimes it slows down so much that using a constraint editor would become impossible at some point. However, even with the really large feature models, managing constraints from the XML file of the model in FeatureIDE is still possible. Therefore FeatureIDE is still an effective tool for realistic feature modeling.

In the analysis part, all the FAMA operations for extended feature models were used in the study. We saw that, using some analysis operations such as Products and Number of Products, takes too much time when the number of features in the model reaches a certain threshold. Faster heuristics for solving constraint satisfaction problems is an active research area. Apart from these, FAMA framework is found useful and flexible for feature model analysis.

The main contribution of this work is the modeling experience gained by researchers and developers working on a commercial product family. Use of model transformations for tool integration is well understood. This work applies model-driven engineering principles to tool integration in the domain of feature modeling.

Most analyses are clearly impossible without using an automated analysis tool like FAMA. These analyses suggested improvements for the model and solutions to fix the errors, and helped the model, proposed solutions to fix the errors, and helped developers to understand model better. Therefore money and time can be saved as a result of the analyses in the early stages of development.

References

[1] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Nowak, and A. S. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," Technical Report, Software Engineering Institute, Carnegie Mellon University, November 1990.

[2] J. Coplien, D. Hoffman, and D. Weiss, "Commonality and variability in software engineering," *IEEE Software*, vol. 15, no. 6, pp. 37-45, November 1998.

[3] M. S. Alencar, *Digital Television Systems*, Cambridge University Press, 2009.

[4] C. A. Poynton, *Digital Video and HDTV: Algorithms and Interfaces*, Morgan Kaufmann Publishers, Amsterdam, Boston, 2003.

[5] H. Benoit, *Digital Television MPEG-1, MPEG-2 and Principles of the DVB System*, Wiley Sons Inc, New York, 1997.

[6] T. Thüm, C. Kästner, F. Benduhn, J. Meinicke, G. Saake, and T. Leich, "FeatureIDE: An extensible framework for feature-oriented software development," *Science of Computer Programming*, vol. 79, pp. 70-85, January 2014.

[7] FAMA Framework. (November 25, 2013). [Online]. Available: http://www.isa.us.es/fama/?Fama_Framework.

[8] Choco-solver. [Online]. Available: http://choco-solver.net/. Accessed November 20, 2013.

[9] G. Arango. Domain analysis methods. In W. Schafer, R. Prieto-Diaz, and M. Matsumoto, editors, *Software Reusability*, chapter 2, pages 17-49. Ellis Horwood, London, UK, 1994.

[10] J. Estublier and G. Vega, "Reuse and variability in large software applications," in *Proc. the 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2005, pp. 316-325.

[11] Creatly. (November 29, 2013). [Online]. Available: http://creately.com/

[12] FAMA user manual. [Online]. Available: http://www.isa.us.es/fama/?Documentation/

Gülseren Fedakar Gönül received her B.S. degree and M.S. degree in computer engineering from the Middle East Technical University (METU) in 2010 and 2013 respectively. Her research is focused on the area of feature modeling, software product line, and variability management on her thesis. She is presently working as a software developer at Social Security Institution of Turkey.

Halit Oğuztüzün is an associate professor in the Department of Computer Engineering at Middle East Technical University (METU), Ankara, Turkey. He obtained his BS and MS degrees in computer engineering from METU in 1982 and 1984, and PhD in computer science from University of Iowa, Iowa City, IA, USA in 1991. His current research interests include model-driven engineering and distributed simulation.

Ahmet Serkan Karataş is a part-time lecturer in the Department of Computer Engineering at Middle East Technical University (METU), Ankara, Turkey. He obtained his BS, MS and PhD degrees in computer engineering from METU in 1998, 2000 and 2010 respectively. His current research interests include variability modeling and management in software product lines.