Modeling Variation in SaaS Application

Eshtiag A. Abd Elrhman [1] Nadir K.Salih [2]
Deanship of Community Service and Continuing Education, Jazan University, Jazan, KSA[1]
Electrical and computer engineering department, college of engineering, Khartoum, Sudan[2]

Abstract
Multi-tenancy gives SaaS application opportunity to realize economical goal for user and provider. For all centralized management there is an important thing to be handling in SaaS environment is the variation of user and system requirements. In this paper we used the feature model in the stage of domain analysis after selected extend that suitable for tenants, which presents the variability for dynamic properties exactly in configuring our model by used algorithm for selecting feature and account variability and commonality. We explained this features in variable homecare model and configured three different scenarios showed the variations in the model. In addition to this, we were adjusting the tenant costing by balancing between variability and commonality. We depend on fuzzy logic to model that relationship. Finally we concluded and remarked for our future work in multi-tenancy SaaS application.
Keywords: Multi-tenancy, SaaS, variability model, feature model.

I. Introduction

To capture and manage commonality and variability the feature model is used in the stage of domain analysis in the context of software product line [1] [2]. Feature is the distinguishing characteristic of an item. It means using feature as the fundamental elements to build a model, in order to form the variation of feature combination. It can be seen as a compact representation of all software products in SPL. Feature models are used for software development process: Firstly, in specification of the requirements, they allow the designers to define which configurations of the software will be supported by the product line. In this stage they give to the designers a view on the entire family of systems. Secondly, in composition of the software they allow the developers to select a particular configuration of the system and so they provide a view on a specific application. Feature model [26] was originally proposed as a Feature Oriented Domain Analysis [28] method in the software engineering approach. The Context Variability Model provides an intuitive way to capture the variability and commonality in the requirements that originate from different contexts, such as different product types, geographic regions and customers. With this concept it is possible to capture several dimensions of variability in the context space in one model [3]. Variability modeling in feature models, that demonstrated the importance of binding time analysis when translating requirements into feature models [4]. Later, feature models play a central role in the development of a system family architecture[25], which has to realize the variation points specified in the feature models [5][27][29]. In application engineering, feature models can drive requirements elicitation and analysis. Knowing which features are available in the software family may help customers to decide which features their system should support. Knowing which desired features are provided by the system family and which have to be custom-developed helps to better estimate the time and cost needed for developing the system. Multi-tenancy SaaS applications depend on tenant that using and working at runtime, according to this we can take just six techniques [9] [10] from Software Product Line that can be suitable for SaaS application:

- Binary Replacement.
- Linker Directives.
- Infrastructure Centered Architecture.
- Run-time Variant Components Specifications.
- Variant Component Implementations.
- Condition on Variable.

Any of above techniques can apply dynamically in runtime to adaptive SaaS applications.

The purpose is to countermeasure the complexity of software systems by making systems self-managing [30] [31] [32] [33] [34] [35] which can decrease costs and enhance the organization's ability to react to change [37]. Self-managing, it must have an automated method to collect and monitor the details it needed from the system; analyze those details to determine if something needs to change; to create a plan, or sequence of actions, that specifies the necessary changes; and to perform execution for those actions. When these functions can be automated, an intelligent control loop is formed. Broad-ranging autonomic solutions require designers to account for a range of end-to-end issues affecting programming models [36].

In a multi-tenancy environment, all clients and their users consume the service from the same technology platform, sharing all resources in the technology stack including the data model, servers, and database layers. Software applications must be specifically built for multi-tenancy. Attempting to add multi-tenancy to an existing application not built for it, is analogous to trying to convert a single-user desktop application to a multi-user Web application. The variation of wishes and requirements of users became very big problem for provider, in order to solve this problem we apply variable model. Multi-tenancy architecture depicted in figure 1 it includes four modules:

Authorization module: very important because any tenant has many users, share of resources and databases. Need security in all layers users, data, and resources.

Server module:
We can say all managements for services in multi-tenancy SaaS responsible from this module.

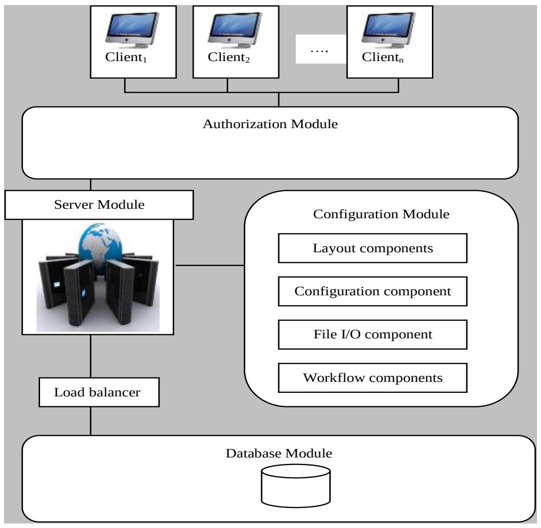


Fig 1 Multi-Tenancy SaaS Architecture

Configuration module:
It shows configuration for multi-tenancy SaaS application variable in runtime for layout maybe change for user or different from user to user, any tenant has profile component configuration, file input and output configuration, and which is the best workflow configuration.

Database module:
All database store in one place. This need query system to make adaptation between tenant and system. In addition load balance can make tuning for storages.

Variation of SaaS system can be taking from different perspectives user level, tenant level variation. For example the variation in tenant level lead to isolation in:

Functional Data can choose either to share the database between tenants or have separate database/schema for each tenant. In general accessibility of data can be tenant level. Behaviour determined by business logic to defined functionalities of application. Multi-tenant application is possibility that certain functionalities can be alter or customized for every tenant. View presentation tier of application each tenant would prefer to have their own color scheme, style, look …etc.

Operational Performance non-functional requirements provided by customers , multi-tenant application is require to provide performance isolation for each tenant's specified SLA. Availability is governed for customer by SLA. In multi-tenant application availability requirement, for application could differ from customer to customer. Security in multi-tenant scenario however security requirements could vary between tenants. Multi-tenant application should provide provision to address these varying customer expectations.

Support variability within multi-tenant SaaS application for controlled what we need to analyze the degree of variability. Requirement of tenants represent services. It must take (Mandatory) or it may take (Optional) .Variability is the differences between functional and nonfunctional property of tenants. Communality is the same feature for all tenants. It is Important to show higher degree of sharing feature by tenants. In this research, we need to balance between variability and commonality for the existing tenants.

Contribution

In this research work, we realized the following contributions: In problem space, first within a SaaS product we offered a different variant to all customers' requirements (such as, show doctor the variation of patient cases) in the domain of analysis, Second in design domain, for multi-tenancy application we applied flexible operations. Third in solution space and implementation we have transferred the model to object oriented language for dynamically configuration application and account variability and commonality. Fourth we used fuzzy logic to adjust tenant costing by balancing between variability and commonality.

The remainder of the paper is organized as follows. Section 2 views Modeling variation in SaaS application. Different configuration of case study is presented in Section 3. Implementation model is described in Section 4. Analysis result of Model system and adjusting tenant cost realized in section 5. Related work showed in section 6. Section 7 gives a conclusions and future work.

II. Modeling variation in SaaS application

We depend on Software Product line to deal with two domains:- Engineering Domain, Application Engineering they have three activities as depicted in figure2:

Analysis Stage: Engineering domain used multiple feature models to represent the functional units, user interaction, and access of data. Can obtain the approximation for variability and commonality by multiple compose the owner feature for any group, Application Engineering, by consideration of commonality and variability feature. We can customize configuration tenants in SaaS application.

Design Stage: Engineering domain make architecture for decision on resolution of quality attributes, to address the quality and functional requirements' for all tenant. Application Engineering, benefit from selection of quality architecture in engineering domain can select a particular architecture for any SaaS application.

Implementation Stage: Engineering domain modularization architecture is to generate all components according estimation between variability and commonality of user requirements. Application Engineering, automatically the SaaS application will generate from component, and select perfect component to realized variation to SaaS application.

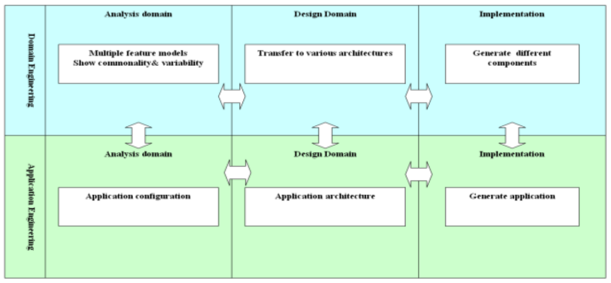


Fig2 Three Stages model

We have taken homecare as example has three cases for different patients: first some patients have diseases lead to dangerous case that is called urgent event the homecare application needs to monitor this case. Second have a kind of patient have diseases just they stay on bed the homecare application will show the doctor the state of patients. Third we want from homecare to monitor any people like diagnose the blood to show if there problem will occur this case called on bathroom. This application will communicated with user by mobile or personal computer through web services.

Analysis Stage

Offer a different variant to all customers' requirements. Homecare system is our context for modeling variation, have three cases, Urgent_event, On_bed , and Bath_room, Communication system for all cases. During Domain Analysis we use the FeaturePlugin to define the feature model, Mandatory must select, Optional can select, not select, Alternative just select only one in one configuration, Or can select one or more. Analysis system appeared in figure3

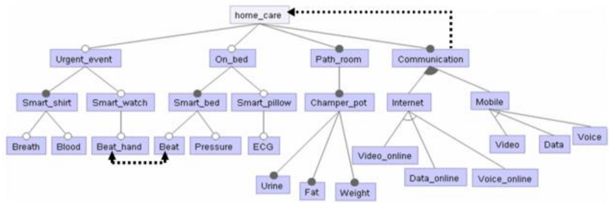


Fig3 Analysis homecare system

Design Stage

In design we have three steps: Step1 Transfer homecare model from feature model to hyper_graph feature model version as showed in figure4

Step2 configure the model nodes and determined the inputs of cases and user requirements as showed in appendix.

Step3 for Modeling SaaS application variation we need flexible operations for change architecture. Composition, we identified three important forms of composition (insert, aggregate, merge).

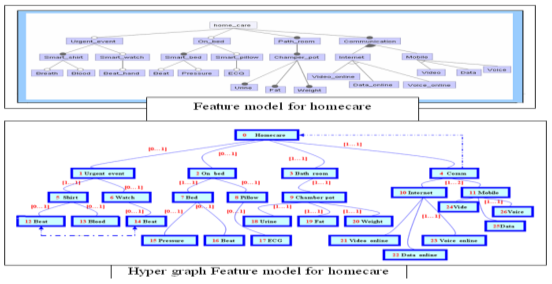

Feature model for homecare
Hyper graph Feature model for homecare

Fig4 transfer feature model to hyper-graph

Insert, aims at introducing new features, already organized in a homecare model, into a specific location of another existing homecare model see figure5

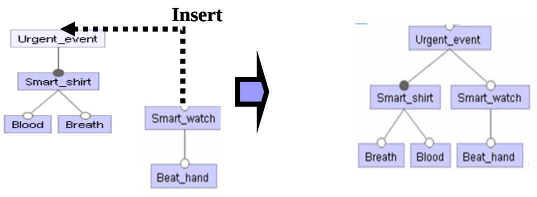


Fig5 insert operation

Aggregate, supports cross-tree constraints between features so that separated homecare models can be inter-related as depict in figure6

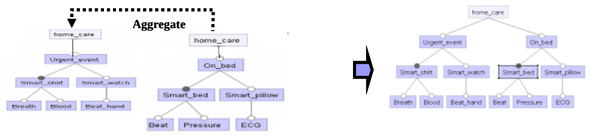


Fig6 Aggregation operation

Merge, is dedicated to the composition of homecare models that exhibit similar features looked at figure7

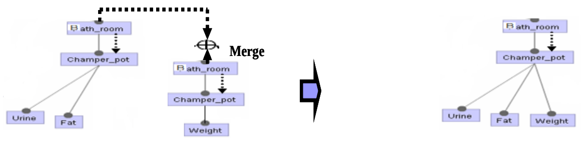


Fig7 Merge operation

Decomposing in figure8 the solutions to the sub-problems is then combined to give a solution to the original problem. Realize flexibility by divided the model into sub variation models to agree with multi-tenant differences.

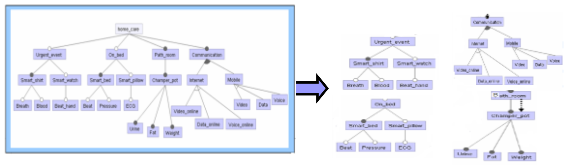


Fig8 Decomposition operation

III Different configuration

Our case study include multi-database [42] for e-health; here just we take homecare as example. During runtime may require reconfigure resources (devices, software) to make dynamic adaptation like the doctor can change care plan at any time this may change devices, relationship between components. We can say the first important thing is what the care plan that interaction between system and patient it similar to workflow, the second thing what is the availability of resources. Configuration process can be depending on patient profile or the Service Level Agreement in clouding environment [43] when it change dynamically obtain new configuration for service as depict in figure9 bellow.

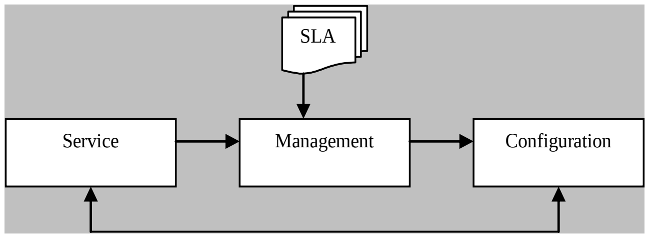


Fig9 Reconfiguration Flow

Configuration of Model system component variation, during runtime may require reconfigure resources (software, devices) to make dynamic adaptation are clear in these scenarios:

Component variation by alternative feature

From feature model of homecare we have a number of alternatives feature; this can be helping the system to use many ways to serve the patient. We can depend on sensor to select what available in communication service if it is included in SLA it can be select if not or the user has change can make reconfiguration to suitable service. We can see in appendix, communication service by internet or mobile, some patient have only used mobile or internet or other have the two options. The system need to make reconfiguration because quality of service is less than the best, this lead to change service to another if it is inside the profile of patient or can add it if the patient make change. We can see the communication system that have two ways with personal computer or mobile and they have alternative feature data can be different format in video, data, and voice. See the new configuration in listing see appendix. It Described availability of nonfunctional property for "communication system". In homecare model have alternative feature for type of data transmission, this help doctor to find patient information in different ways (mean high availability). Reconfiguration depend on demand for example doctor want to see the patient and in his profile didn't found video feature the system can make reconfigure and add this feature the result is service qualify increase. Sometime any alternative fail can change to another. This scenario describes availability nonfunctional property for communication system.

Depend on care plan used optional feature, describe dynamic variability for urgent event that lead to new configuration at runtime. Like in homecare model we have optional feature in case of urgent_event, if occur any problem in smart_shirt can change to smart_watch without stop system. This scenario used care plan that include the workflow of service for any patient and resource used in normal case. Sometimes need to reconfiguration for architecture to change resource depend on optional feature? We depict this case in urgent event that appear in appendix if the patient is used smart shirt to collect information (blood, breath, beat) that will help system to save the patient. In care plan we have other optional feature to measure this information by smart watch, this point need system make new configuration to change the resource that capture information automatically instead of smart shirt because it has not clear measurement. This scenario can describe dynamic variability for urgent event that lead to new configuration at runtime for healthcare system appeared in listing see appendix.

New event for patient

This scenario can make adaptation service sensitive to any changes that may be made by the doctor. Like figures in appendix add new case to the patient of urgent_event, he has two cases. The configuration support framework continuously monitors health problems described in the patient care plan, making the adaptation service sensitive to any changes that may be made by the doctor. If we take example say the patient have disease that monitor from one case we mention in figure 2 like in bed and the profile of patient and care plan include all resources and configuration that can be occur. By monitoring from doctor some time predict some cases may occur for this patient, new disease or we can take urgent event case this will need new configuration. The system modifying in care plan by used new resources that will monitor new case. As we mention the doctor will receive information from smart shirt or watch beside from smart bed or smart pillow as. Here in runtime we have new care plan by adding new configuration for new resources in this case. The doctor wants to monitor on bed case and urgent event in the same time only for one patient. The configuration changed as in listing look at appendix.

IV Implementation model

Implementation for model consisted from two properties: Selected feature that are found as configuration for tenant. Dynamically account, approximation for commonality and variability.

$$T = (T_1, T_2 \ldots\ldots T_n)$$

Valid Configuration G = (VG, EG, r) is a sub-hyper-graph
V_G is a subset of nodes of V: $V_G \subseteq V$
E_G is a set of hyperarcs: $E_G = \{e_G \mid \exists\ e \in E \wedge t(e_G) = t(e) \wedge h(e_G) \subseteq h(e)\}$
The root is present: $r \in V_G$

Algorithm logic

Select feature for any tenant according to three cases:
C1: Select alternative feature mean availability feature (Qos better).
C2: Select optional feature mean dynamic variability (easy to change in runtime).
C3: New event mean adaptation service to any change
Given a hyper-arc,e, with a multiplicity value, mv = [min…max], whose tail (feature) is selected, no less than min and no more than max features of the hyper-arc's head (child features) should also be present in the configuration.
When |H(e)| = 1 (children's cardinality set is one):
- If min = 1 = max, the feature is mandatory, and should present if the parent, or it is a require constraint and the child should also be present [1..1].
- If min = 0 max = 1, the feature is optional [0..1]

When |H(e)| > 1 (children's cardinality set is more than one):
- if min = 1 = max, it is a XOR alternative feature group, and only one of the children should be present at most if the parent is present.
- if min = 0 max = 1, it is an optional feature group, and child features can be present or not as long as its parent is present, or it is a mutex constraint and at most one of the child features can be present
- if $1 \leq$ min $\leq$ max $\leq$ |H(e)|, it is a OR feature group, and no more than max and no less than min child features can be present if the parent feature is present

The figure10 below the input and output of the algorithm. And depict three levels for SaaS application.

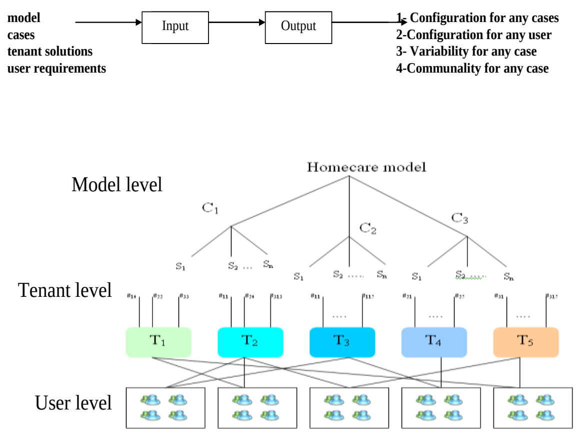


Fig10 input and output of algorithm

Algorithm of selecting feature, and calculation of variability and commonality, and configurations of for used, cases realized in listing1.

```
Algorithm Select_ feature
  Inputs
        Node : N – nodes of all model, Relation: R relationship between nodes, Group : all item belong to any nodes
        C : Cases {$c_1$, $c_2$, $c_3$} :Case1: c1, Case2 :c2, Case3: c3
        T : tenants {$T_1$,$T_2$,$T_3$,$T_4$,$T_5$}:T1 configuration : $T_1$C,$T_2$ configuration : T2C,T3 configuration : $T_3$C, $T_4$ configuration : $T_4$C, $T_5$ configuration : $T_5$C, User_Requirement : UR
  Outputs
        All configuration for any cases : AC,Suitable configuration or any user: SCU
        Account of variability for any case: AV, Account of commonality for any case: AComm
  For each c$\in$ C
      While H(e) > 1 do
            if min = 1 and max=1 Then only one node will select in one configuration ---alternative
            if min = 0 and max= 1 Then in configuration can select or not —optional or mutex constraint
            if 1 min max |H(e)| Then will select all or apart of nodes ---OR
            end if
            end if
            end if
      While H(e) = 1 do
              if min = 1 and max= 1 then must select in configuration —mandatory or require constraint
              if min = 0 and max= 1 then may select or not — optional
              end if
              end if
            configure(c)
            return number of product(k)
          return number of nodes in any case (n)
          AV = $\frac{k}{2^n - 1}$
            AComm = number of appear nodes in all product/k
        end while
        end while
end for each
          if H(e) = 0
            invalid configuration
          for each t $\in$ T
              select SCU configuration for any user
          end for each
end procedure
```

Listing1 algorithm of selecting feature

V Analysis of Model System

We Analyzed Model SaaS application variation by defined Process for the automated analysis of feature models in figure11

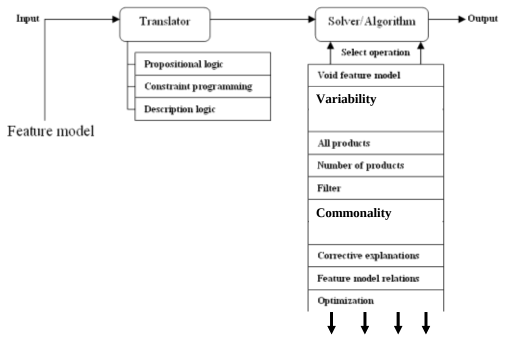


Fig11 process of automated analysis

Variability and commonality for tenants obtained from number of product: This operation takes as input a feature model and returns the number of products of a feature model. This operation reveals information about the flexibility and complexity of the software product line a big number of potential products can reveal a more flexible as well as more complex product line.

$$\text{Variability} = \frac{k}{2^{n}-1}$$

k: is number of products

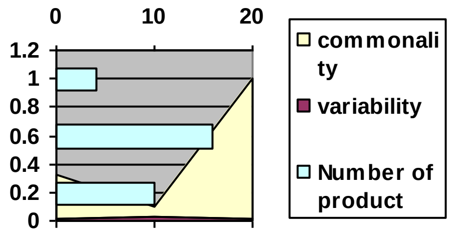
n: is number of all features

Variability increase number of tenant and costing

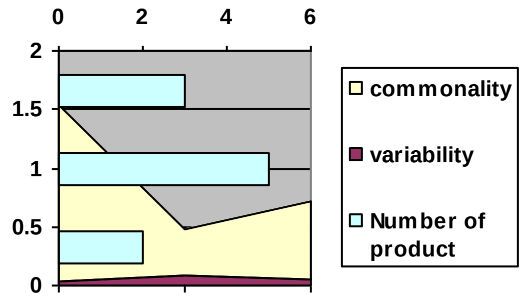
$$\text{Commonality} = \frac{sharednode}{k}$$

Sharednode: number of appeared nodes in all products. Commonality mean reduces number of tenants and costing increase number of product increase number of tenants for that the variation of tenant level can be realized by balancing between variability and commonality as we observed from analysis in three cases of homecare model see bellow tables and figures for three cases.

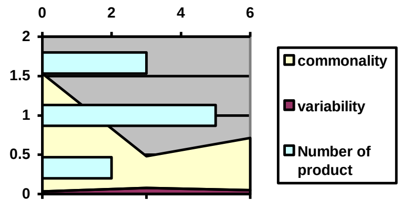
Case1			
variability	0.00782	0.01956	0.0313
commonality	1	0.3	0.0625
Number of product	4	10	16



Case2			
variability	0.031	0.04761	0.0793
commonality	1.5	0.666	0.4
Number of product	2	3	5



Case3			
variability	0.001465	0.00242	0.00366
commonality	0.666	0.4	0.266
Number of product	6	10	15



Adjusting tenants

Balancing between commonality and variability lead to large income for SaaS provider, good services for users economically used of resources. From analysis of three cases of homecare model we observed that relationship

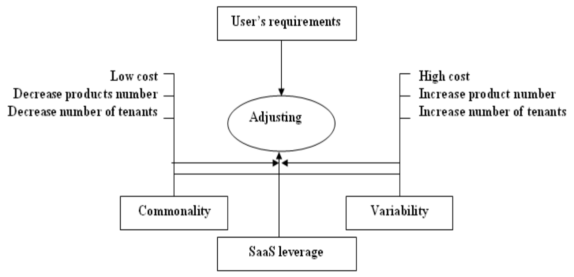


Fig12 adjusting of tenant costing

Adjusting tenants using Fuzzy Expert System

How to obtain the approximation between commonality and variability used fuzzy logic for truth degrees as mathematical model of vagueness phenomenon while probability is mathematical model of ignorance. To make fuzzy controller modeling we looked for three steps:

Fuzzification is proposed observations are uncertainties, first define the input and output, the product number of tenants (δ) calculating from variability (V) and commonality (C). we can set the input as

$$k = [-a, a]$$

$$\bar{k} = [-b, b]$$

Out put set is

$$(\delta) = [-c, c]$$

Fuzzification function is

$$f_k = [-a, a] \rightarrow R$$

R: set of all fuzzy number and $f_k(x_0)$ is fuzzy chosen by f_k As approximations of measurement $k = x_0$. fuzzification by showing the membership function for variability (V) and commonality (C) with trapezoidal shape in figure13 and figure14 respectively.

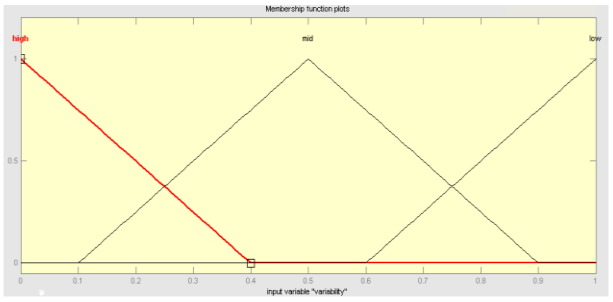


Fig13 Variability membership function

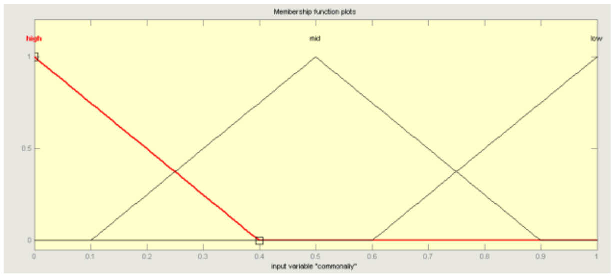


Fig14 Commonality membership function

Fuzzy inference

In our model k, $\bar{k}$ are in put, δ is output:

if k = A, $\bar{k}$ = B THEN δ = C,

A, B, C are fuzzy number to find fuzzy rules we used a set of input, output data:-

$$X\{X_d, Y_d, Z_d \mid d \in D\}.$$

Z_d : output variable of δ

X_d, Y_d : input variable k, $\bar{k}$

If $A(x_d)$, $B(y_d)$, $C(z_d)$ largest membership grades there the degree of relevance is:

$i_1\,[i_2,(\,A(x_d), B(y_d), C(z_d)\,)]$

i_1 ,i_2 are t-norms

fuzzy rules base consist of n fuzzy inference value:

Rule1 if (k, $\bar{k}$) is A_1,B_1, THEN δ is C_1

Rule2 if (k, $\bar{k}$) is A_2,B_2, THEN δ is C_2

Rule_n if (k, $\bar{k}$) is A_n,B_n, THEN δ is C_n

The result of The rule in our model in figure15 is:

If V is high, C is low then costing is high
If V is mid, C is mid then costing is mid
If V is low, C is high then costing is low

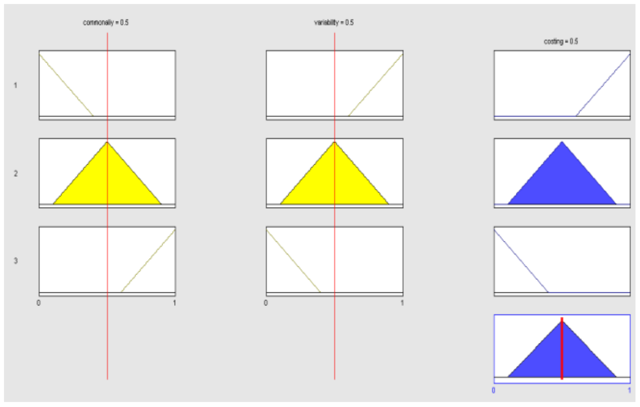


Fig15 Result of Inference Rule

Defuzzified

For calculating Defuzzified number we used centroid method. To convert the out put values inference engine express as fuzzy set. Defuzzified output variable express by :

$$X = \frac{\int_a^b \mu_a(x)\,x\,dx}{\int_a^b \mu_a(x)\,dx}$$

$\mu_a(x)$: Membership function, aggregate membership function to:

1- X_{min} minimum costing(δ)
2- X_{mid} mid costing(δ)
3- X_{max} maximum costing(δ)

All the rules have been depicted as 3D graphs called surface viewer in Figure16:

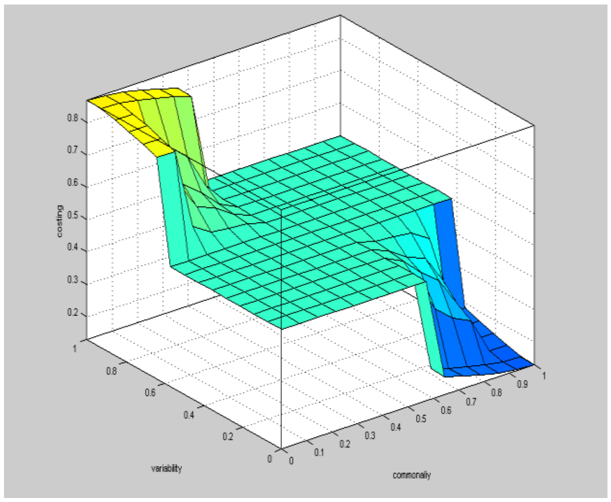


Fig16 Surface for Commonality, Variability and Cost

VI. Related work

1. Software Product Line

C. Cetina et al in [11] they used feature model at runtime to enable smart home system perform reconfiguration like query feature model to change in architecture. Kyo C. Kang et al in [12] they described The Feature-Oriented Reuse Method concentrates on analyzing and modeling a product line's commonalities and differences in terms of features and uses this analysis to develop architectures and components. The FORM explores analysis and design issues from a marketing perspective. Mohammad and Hassan in [13] they used feature model to model SOA variability for maximizing reusability, and allow service providers and consumers to change independently of each other, since all variability is performed at the service contract or service interface level only. Mathieu Acher et al in [14] they made interaction between specification and Implementation by choice feature model rules to improved configuration and adaptation runtime. Liwei Shen et al in [15] they used feature models to capture runtime variations to implement runtime reconfigurations and implementation-level code adaptation adopting dynamic. Hisayuki Horikoshi et al in [16] they depend on a feature-oriented analysis technique to identify adaptation points, and calculate the contribution to non-functional goals of the configuration a component specification model, which extends an architectural description language for self-adaptation and reduced reconfiguration at runtime.

All authors in this related work used the feature model as variability model in SPL not like us we used it in multi-tenancy SaaS.

2. Multi-tenant SaaS

Ali Ghaddar et al. in [17] they apply variability concept in application layer by made variability in model to represent application variation. They used variability in this system to enhance its availability and adaptation to different tenants. R. Mietzner et al. [18] they using explicit variability models to systematically derive customization and deployment information for individual SaaS tenants. Attract a significant number of tenants, to be customizable to fulfill the varying functional and quality requirements of individual tenants. J. Schroeter et al, in [19] they identify requirements for such runtime architecture and they extended existing architecture for dynamically adaptive applications for the development and operation of multi-tenant applications. J. Kabbedijk et al, in [20], they design three architectures design pattern for variability in multi-tenant environment. K.ozturk et al in [21] they provide a feature model for SaaS that depicts the design space and represents the common and variant parts of SaaS architectures. R.Mietzner et al, in [22] they show how the service component architecture (SCA) can be extended with variability descriptors and SaaS multi-tenancy patterns to package and deploy multi-tenant aware configurable composite SaaS applications. H. Jung et al in [23] they proposed process, SaaS services with high quality can be effectively developed. And highlight the essentiality of commonality and variability (C&V) modeling to maximize the reusability. R.Mietzner et al in [24] they describe the notion of a variability descriptor that defines variability points for the process layer and related artifacts of process-based, service-oriented SaaS applications. Here all the above related works are talking about variability in multi-tenancy SaaS but didn't talk about configuration and calculation of commonality, variability dynamically.

VII. Conclusion and Future Work

In this work we proposed variability model that realize the dynamic properties to help provider in multi-tenancy SaaS application manage variation in user and system requirements. The feature model help us determined the alternative and optional variation in our homecare model. Automatically the changed occur in configuration appear in XML file because we design this model in eclipse platform. Dynamically we selected feature and calculation variability and commonality. In addition we adjusting tenant costing by balancing between variability and commonality.

For future work, we intend to extend our model to support more aspects of SaaS application development. Specifically, we are planning to extend the model for metrics management, detailed software configuration aspects, integrate a model for multi-tenant architecture, and establish mechanisms to link all these artifacts to source code.

References

[1] Frank van der Linden, Klaus Schmid and Eelco Rommes, Software Product Lines in Action, Springer-Verlag Berlin Heideberg, 2007.
[2] M. Acher, P. Collet, R. B. France.Separation of Concerns in Feature Modeling: Support and Applications. https://nyx.unice.fr/publis/acher-collet-etal:2012.pdf
[3] H. Hartmann, T. Trew. Using Feature diagrams with Context Variability to model Multiple Product Lines for Software Supply Chains.IEEE, 2008.

[4] LAU, SEAN Q: Domain Analysis of E Commerce Systems Using Feature-Based Model Templates. MASc Thesis. University Avenue West, Waterloo, Ontario, Canada, 2006.
[5] K. Czarnecki, S. Helsen, U.Eisenecker. Staged Configuration Using Feature Models, pp. 266–283, Springer-Verlag Berlin Heidelberg, 2004.
[6] G. Gallo, G.Longo, S. Nguyen, S. Pallottino, Directed Hyper Graphs and Applications. http://www.cis.upenn.edu/~lhuang3/wpe2/papers/gallo92directed.pdf.
[7] P.Yves Schobbens, P. Heymans, J.Christophe Trigaux, Y. Bontemps. Generic semantics of feature diagrams. Pp:456–479, Elsevier,2007.
[8] Miguel A. Laguna, José M. Marqués. Feature Diagrams: a Formalization and extensible Meta-model Proposals. http://giro.infor.uva.es/TR2009-2.pdf.
[9] S. T. Ruehl, U. Andelfinger. Applying Software Product Lines to create Customizable Software-as-a-Service Applications. ACM, 2011.
[10] M. Svahnberg1, J. van Gurp, Jan Bosch. A taxonomy of variability realization techniques. Wiley InterScience, 2005.
[11] C. Cetina, P. Giner, J. Fons, V. Pelechano.
Autonomic Computing Through Reuse OF Variability Models at Runtime: The Case of Smart Homes. IEEE, 2009.
[12] K. C. Kang, J. Lee, P. Donohoe. Feature Oriented Product Line Engineering. IEEE, 2002.
[13] M. Abu-Matar. H. Gomaa. Variability Modeling for Service Oriented Product Line Architectures. International Software Product Line Conference. IEEE, 2011.
[14] M. Acher, P. Collet, P. Lahire, S. Moisan, J.-Paul Rigault. Modeling Variability from Requirements to Runtime. International Conference on Engineering of Complex Computer Systems.IEEE. 2011.
[15] L. Shen, X. Peng, J. Liu , W. Zhao. Towards Feature-oriented Variability Reconfiguration in Dynamic Software Product Lines. http://www.se.fudan.edu.cn/paper/ourpapers/204.pdf
[16] H. Horikoshi, H. Nakagawa, Y. Tahara, A. Ohsuga. Dynamic Reconfiguration in Selfadaptive
SystemsConsidering Nonfunctional Properties. ACM, 2011.
[17] Ali Ghaddar, Dalila Tamzalit, Ali Assaf. Decoupling variability management in multi-tenant SaaS applications. International Symposium on Service Oriented System Engineering, IEEE.2011.
[18] Ralph Mietzner, Andreas Metzger, Frank Leymann, Klaus Pohl. Variability Modeling to Support Customization and Deployment of Multi-Tenant-Aware Software as a Service Applications. ICSE'09 Workshop, IEEE, 2009.
[19] Julia Schroeter, Sebastian Cech, Sebastian Gotz, Claas Wilke, Uwe Abmann. Towards Modeling a Variable Architecture for Multi-Tenant SaaS-Applications. Sixth International Workshop on Variability Modeling of Software-Intensive Systems.ACM, 2012.
[20] Jaap Kabbedijk, Slinger Jansen. Variability in Multi-tenant Environments: Architectural Design Patterns from Industry, Springer, 2011.
[21] K. Ozturk, B. Tekinerdogan, Feature Modeling of Software as a Service Domain to Support Application Architecture Design. International Conference on Software Engineering Advances. IARIA, 2011.
[22] Ralph Mietzner, Frank Leymann, Mike P. Papazoglou. Defining Composite Configurable SaaS Application Packages Using SCA, Variability Descriptors and Multi-Tenancy Patterns. Third International Conference on Internet and Web Applications and Services, IEEE, 2008.
[23] Hyun Jung La, Soo Dong Kim. A Systematic Process for Developing High Quality SaaS Cloud Services. Springer, 2009.
[24] Ralph Mietzner, Frank Leymann. Generation of BPEL Customization Processes for SaaS Applications from Variability Descriptors, International Conference on Services Computing.IEEE, 2008.
[25] M. Moon, H. S. Chae, K. Yeom. A Meta-model Approach to Architecture Variability in a Product Line. pp. 115 – 126,Springer, 2006.
[26] Don Batory. Feature Models, Grammars, and Propositional Formulas. pp. 7 – 20, Springer-Verlag Berlin Heidelberg, 2005.
[27] M. Asadi, B.Mohabbati, D. Gasevic, E.Bagheri2, M. Hatala. Developing semantically-enabled Families of MethodorientedArchitectures. http://ebagheri.athabascau.ca/papers/ijismd.pdf
[28] K. Kang. Feature-oriented domain analysis (FODA) feasibility study. Technical report, DTIC Document, 1990.
[29] Luca Gherardi, Davide Brugali. An eclipse-based Feature Models toolchain. http://www.best-ofrobotics.org/pages/publications/UniBergamo_EclipseIT2011.pdf
[30] P. Horn. Autonomic computing: Ibm's perspective on the state of information technology, 2001.
[31] S. Pandey, W. Voorsluys, S. Niu, A. Khandoker, R. Buyya. An autonomic cloud environment for hosting ECG data analysis services. Elsevier, 2011.
[32] E. Arnautovic, H. Kaindl. Towards Self-Managed Systems aware of Economic Value. International Conference on Self-Adaptive and Self-Organizing Systems Workshop, IEEE, 2010.
[33] Alexandra Carpen. Towards a Self-Adaptive Data Management System for Cloud Environments. International Parallel & Distributed Processing Symposium, IEEE, 2011.
[34] M.Rahman, R. Ranjan, R. Buyya, B. Benatallah. A taxonomy and survey on autonomic management of applications in grid computing environments. John Wiley & Sons, Ltd. 2011.
[35] G. Blair, N. Bencomo, Robert B. France. Models@ Run-Time. IEEE, 2009.
[36] S. Dobson, S. Denazis, A. Fern´andez, D. Gaiti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli. A survey of autonomic communications. ACM Transactions on Autonomous and Adaptive Systems, 1:223–259, 2006.
[37] IBM. An architectural blueprint for autonomic computing. Technical report, IBM., 2006.
http://people.cs.kuleuven.be/~danny.weyns/csds/IBM06.pdf
[38] D, Valtchev, I. Frankov. Service Gateway Architecture for a Smart Home. IEEE, 2002.
[39] Gerald Tesauro. Reinforcement Learning in Autonomic Computing a Manifesto and Case Studies. IEEE Computer Society, 2007.
[40] D. Benavides, S. Segura, P. Trinidad, A. R.Cortes. FAMA: Tooling a Framework for the Automated Analysis of Feature Models.
[41] R. Gupta, S. K. Malik. SPARQL Semantics and Execution Analysis in Semantic Web Using Various Tools. IEEE, 2011.
[42] Nadir K Salih, Tianyi Zang, Mingrui Sun.Multi databases in Health Care Networks. International Journal of Computer Science Issues, vol. 8, Issue6, No3, 2011, pp 210-214.
[43] Nadir K Salih, Tianyi Zang. Survey and comparison for Open and closed sources in cloud Computing. International Journal of Computer Science Issues, vol. 9, Issue 3, No1, 2012, pp 118-123.