Towards Event Sequence Graph-based Testing of Feature-oriented Software

Fevzi Belli

Tuğkan Tuğlular

Dilek Öztürk (presenter)

Preliminaries

How to define Feature-oriented Software?

Software products that have common features. These products vary with their different features.

Preliminaries

What is the best way to develop Feature-oriented software?

Exploiting software reuse to develop the variants that have common features.

Preliminaries

Developing feature-oriented software by taking advantage of software reuse is risky because:

One reusable component fits one product variant perfectly where it casuses severe faults within another. These products become poor quality.

Preliminaries

How to assure quality of the variants?

Verifying and validating each variant.

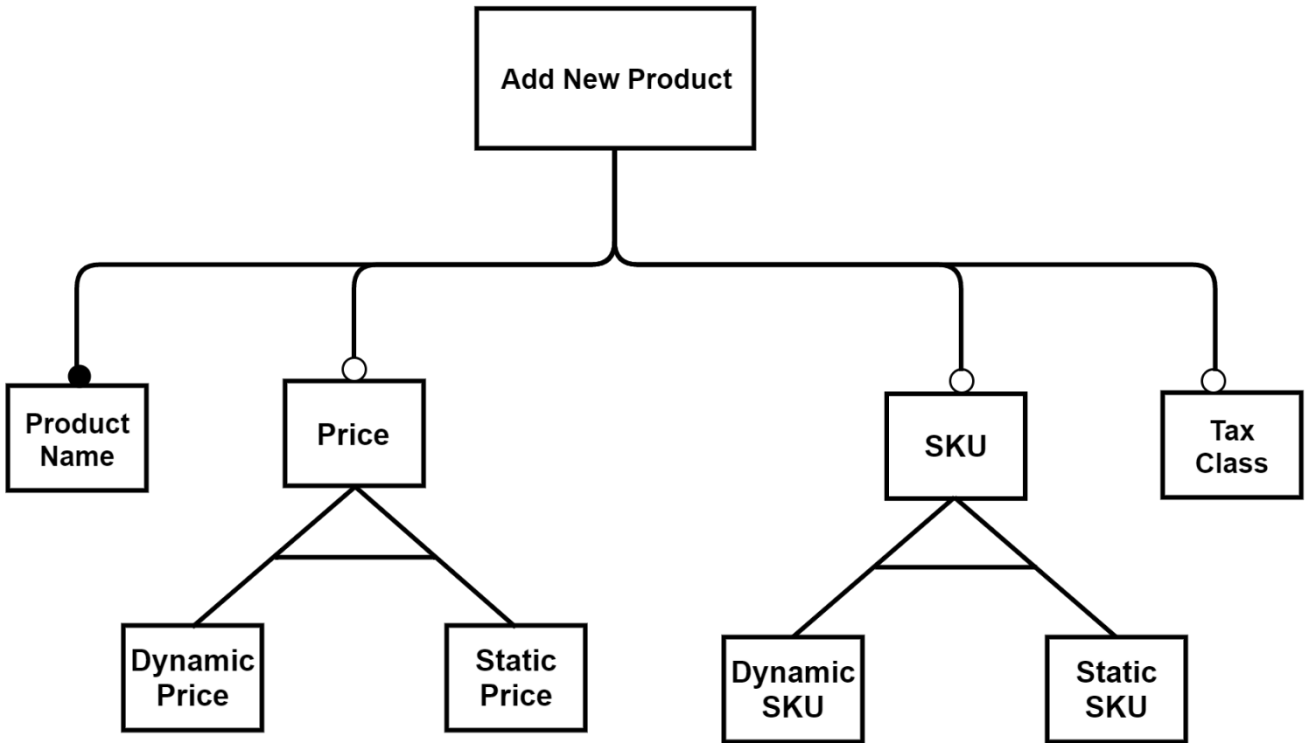
Main Objective

To suggest an approach to systematically test potentially very large number of product variants.

In order to achieve this, coupling feature diagrams with event sequence diagrams for testing purposes.

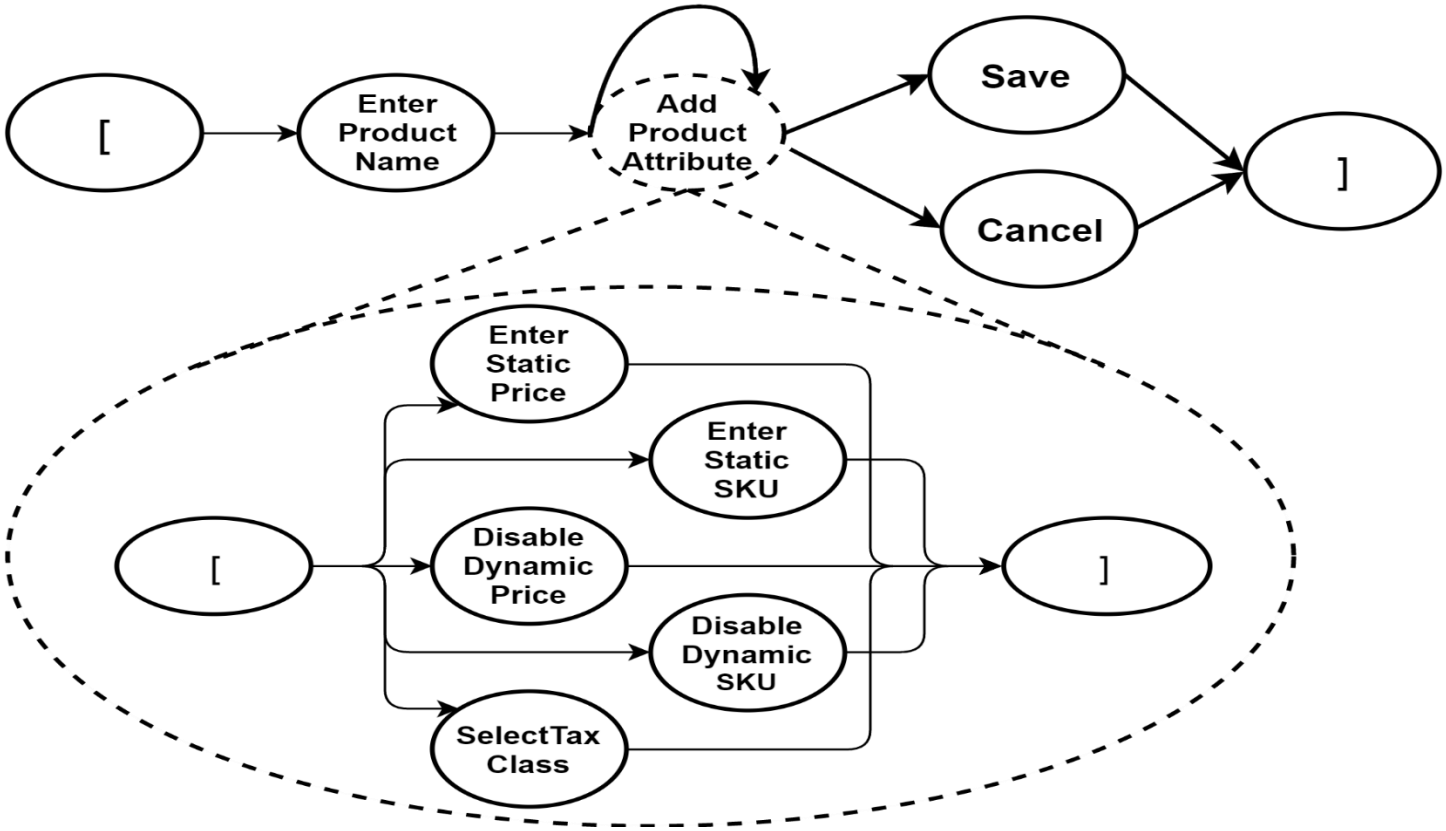
Terms: Feature Diagram

Feature Model: **Indicative model of variation points among products.**



Terms: Event Sequence Graph

Event Sequence Graph: **Models the interactions between a system and its user. It is used as a test-generative model.**

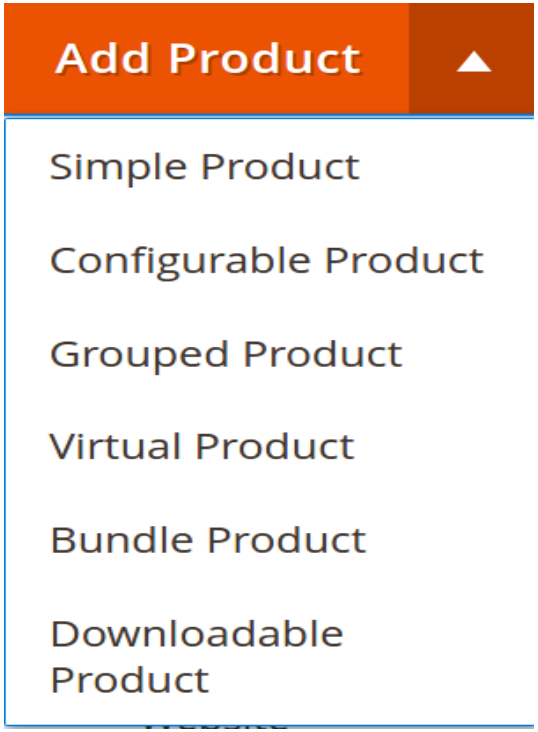


Running Example

MAGENTO: an e-trade software that has modules resembling feature-oriented software product sets

Running Example

Magento's *Add New Product* module has six product variants that form a feature-oriented product set.



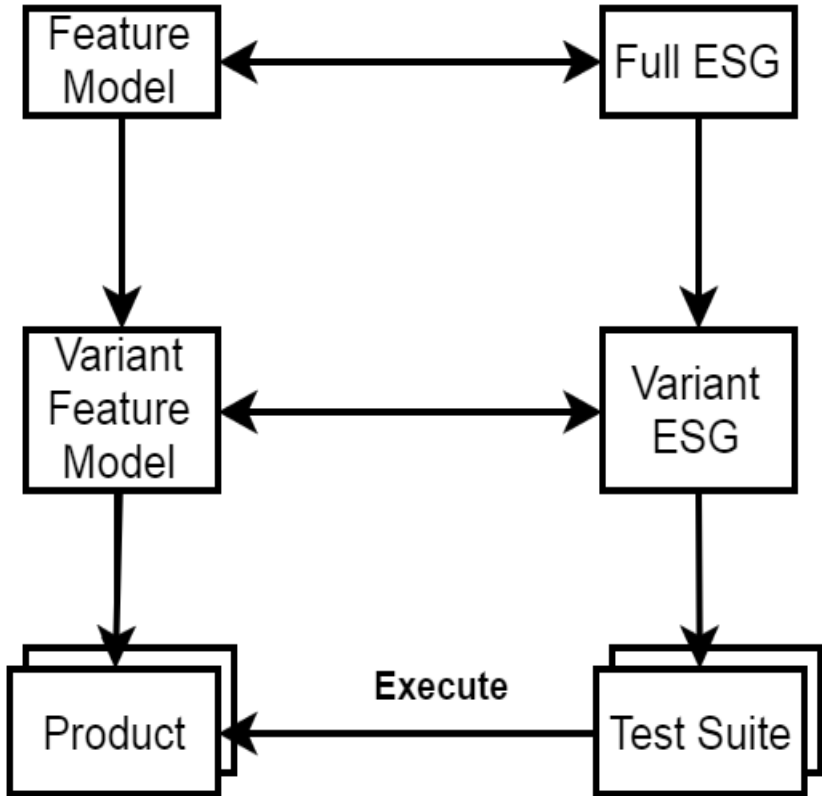
Simple, Bundle and **Downloadable** has been selected in this study.

Running Example

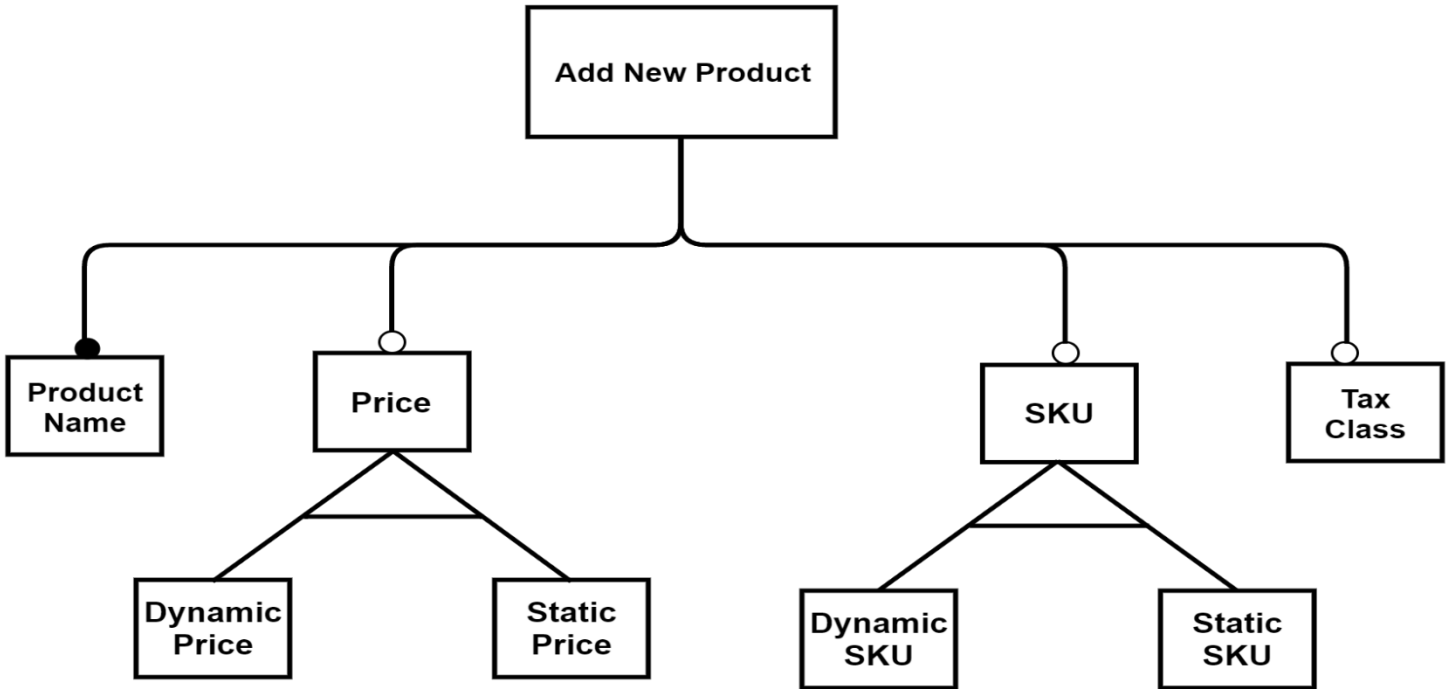
	Simple Product	Bundle Product	Downloadable Product
Product Name	✓	✓	✓
Static SKU	✓		✓
Dynamic SKU		✓	
Static Price	✓		
Dynamic Price		✓	✓
Tax Class	✓		✓

Approach

- A feature model is built to indicate variation points among products.
- A *full*-ESG which represents overall system behavior within product set is constructed.
- The Feature Model and the ESG are coupled.
- A *variant*-ESG is derived.
- Positive and negative test cases are generated from variant-ESGs.

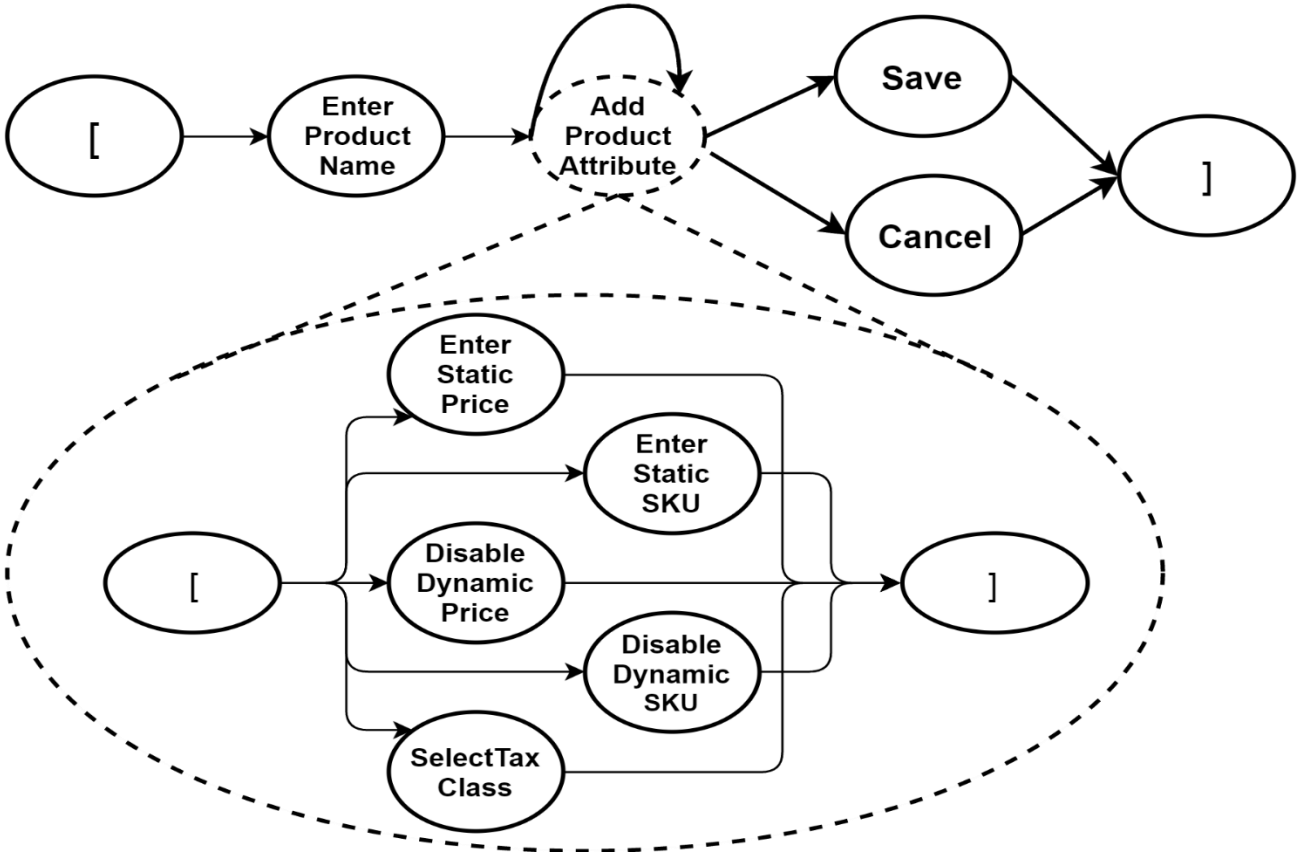


Indicative Example



Feature Model

Indicative Example

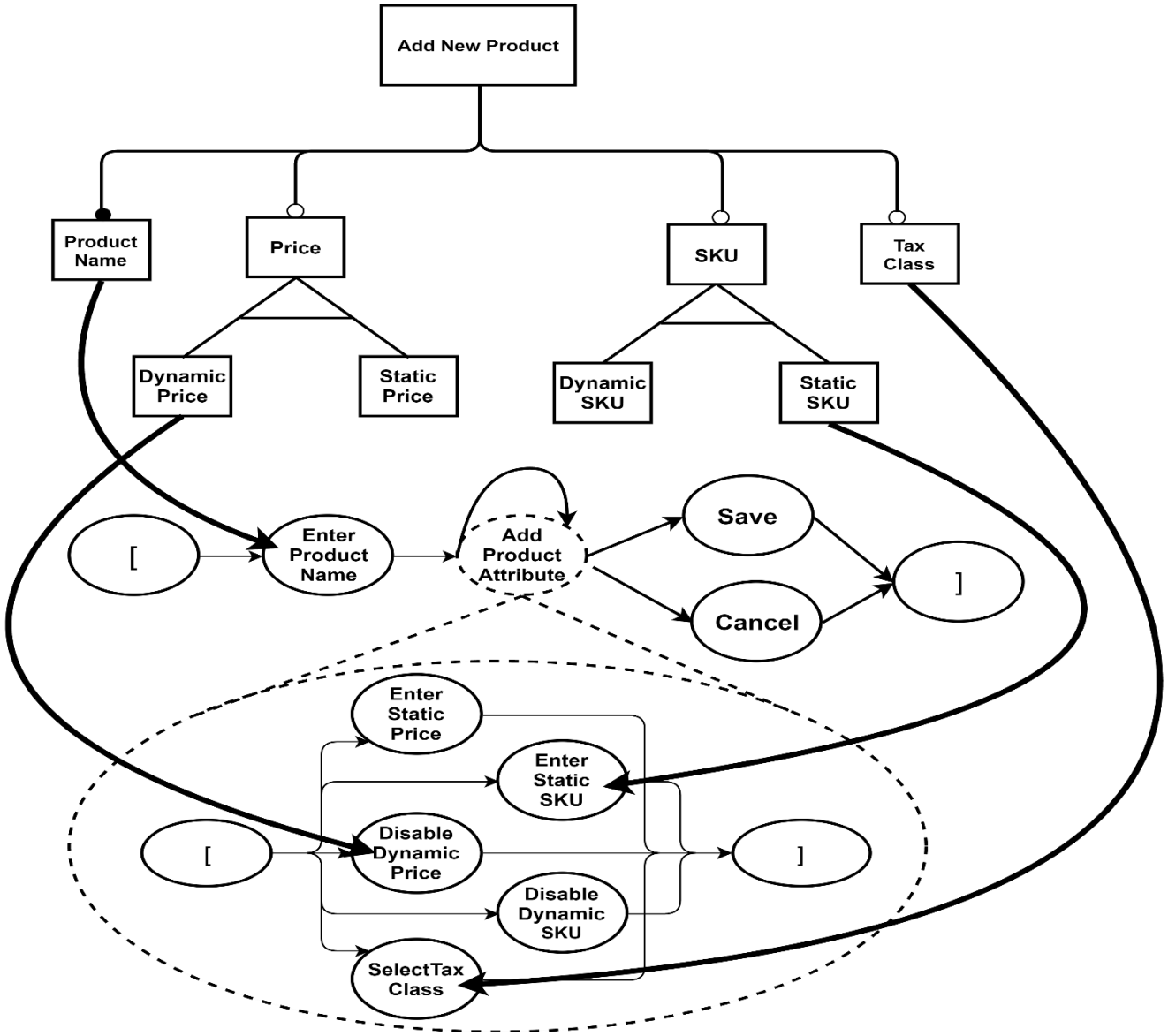


full-ESG

Indicative Example

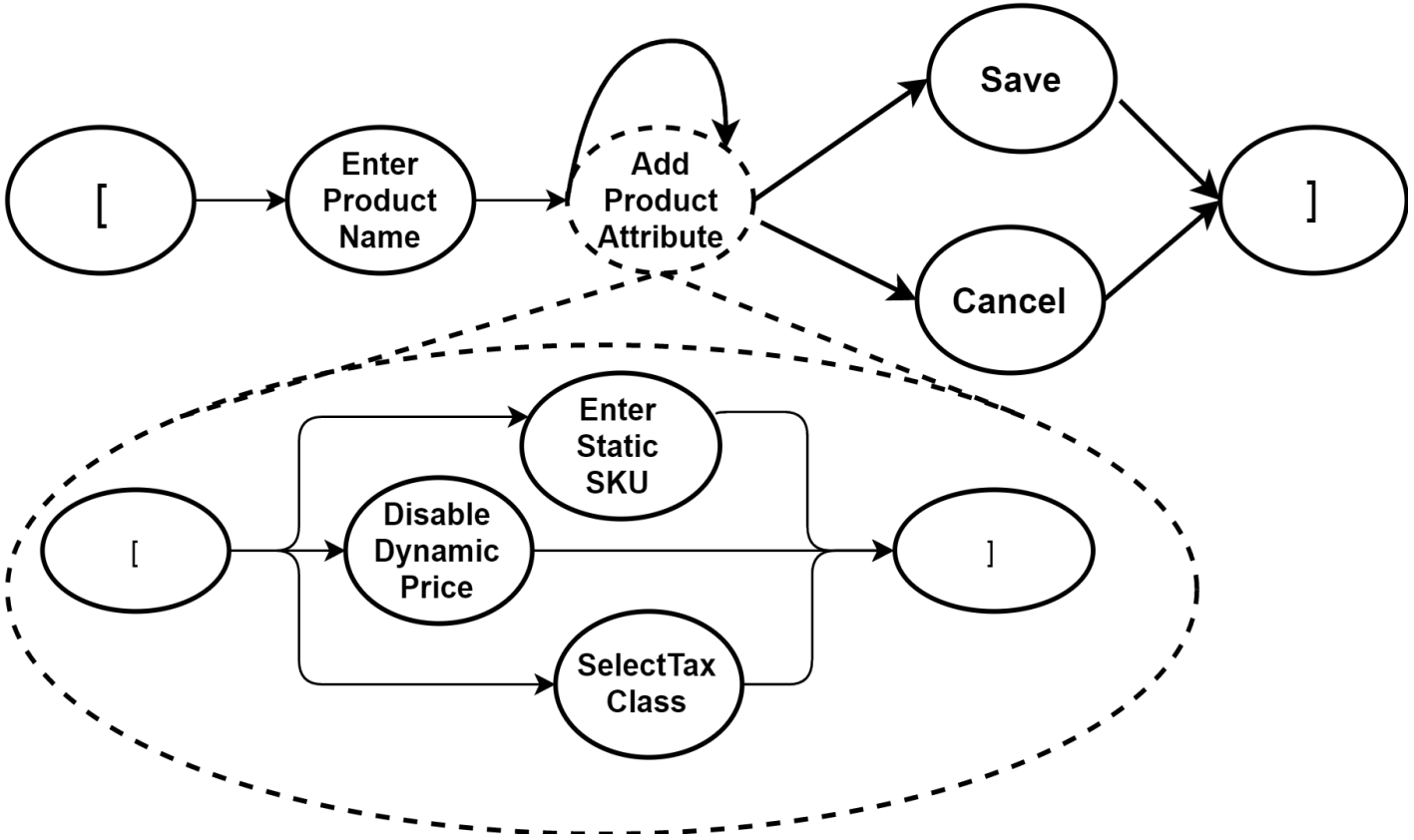
	Simple Product	Bundle Product	Downloadable Product
Product Name	✓	✓	✓
Static SKU	✓		✓
Dynamic SKU		✓	
Static Price	✓		
Dynamic Price		✓	✓
Tax Class	✓		✓

Indicative Example



Coupling of Feature Diagrams with Event Sequence Graphs

Indicative Example



Variant-ESG: Downloadable Product

Indicative Example

	Simple Product	Bundle Product	Downloadable Product
Product Name	✓	✓	✓
Static SKU	✓		✓
Dynamic SKU		✓	
Static Price	✓		
Dynamic Price		✓	✓
Tax Class	✓		✓

Indicative Example

12: [, EnterProductName, EnterStaticSKU, EnterStaticSKU, DisableDynamicPrice, DisableDynamicPrice, SelectTaxClass, SelectTaxClass, DisableDynamicPrice, EnterStaticSKU, SelectTaxClass, EnterStaticSKU, Save,]
3: [, EnterProductName, DisableDynamicPrice, Save,]
3: [, EnterProductName, SelectTaxClass, Save,]
3: [, EnterProductName, SelectTaxClass, Cancel,]
3: [, EnterProductName, EnterStaticSKU, Cancel,]
3: [, EnterProductName, DisableDynamicPrice, Cancel,]

The six positive and twenty three negative test cases form a test suite for Downloadable Product.

Conclusion

RISK OF SOFTWARE REUSE

One reusable component fits perfectly to one variant whereas it causes severe faults for another variant

Conclusion

To model variations and commonalities among products **FEATURE MODEL**

To generate test cases automatically **EVENT SEQUENCE GRAPHS**

Conclusion

OBJECTIVE

Building an automated testing for large sets of feature-oriented software products.

APPROACH

Feature Diagrams and Event Sequence Graphs are coupled.

Possible Impacts of this study

- the productivity of companies
- quality of individual products
- percentage of component reuse
- the return on investments

INCREASES

- the cost
- the labor needs
- the time to release a product

DECREASES

References

[1] Weißleder, S. Lackner, H. (2013). Top-Down and Bottom-Up Approach for Model-Based Testing of Product Lines. MBT, volume 111 of EPTCS, page 82-94. Electronic Proceedings in Theoretical Computer Science (EPTCS) 111, pp. 82-94.

[2] Utting, M. Legeard, B. (2006). Practical Model-Based Testing: A Tools Approach. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

[3] Belli, F., Linschulte (2007), M., On 'Negative' Tests of Web Applications, Annals of Mathematics, Computing & Teleinformatics, vol. 1, vol. 5, pp. 44-56.

[4] Magento. (n.d). Retrieved April,8 2017 from https://magento.com/products/community-edition

[5] TestSuiteDesigner. (n.d, Univ. of Paderborn, Angew. Datemtechnik). Retrieved April,8 20017 from http://download.ivknet.de/ .

[6] Lochau, M. Oster, S. Goltz, U. Schürr, A. (2012) Model-based pairwise testing for feature interaction coverage in software product line engineering. Software Quality Journal 20(3-4), pp. 567–604, doi:10.1007/s11219-011-9165-4.

[7] Carnegie Mellon University (2012): Software Product Lines. Retrieved April,12 2017 from http://www.sei.cmu.edu/productlines/

[8] Belli, F., Budnik, Ch. J., White, L. (2006), Event-based Modeling, Analysis and Testing of User Interactions: Approach and Case Study, Software Testing, Verification and Reliability, vol. 16, 1, 3-32, John Wiley & Sons, Ltd.

[9] Kang, K.C. and Cohen, S.G. and Hess, J.A. and Novak, W.E. and Peterson, A.S. (1990), Feature-oriented domain analysis (FODA) feasibility study, Technical Report CMU/SEI-90-TR-021, SEI, Carnegie Mellon University.

[10] D.B. West (1996), Introduction to Graph Theory, Prentice Hall.

[11] the balance: What Is a Stock Keeping Unit (SKU)? Retrieved May, 28 2017 from https://www.thebalance.com/what-is-a-sku-in-retail-terms-2890158