Variability Management in an unaware software product line company. An experience report

David Benavides and José A. Galindo
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Sevilla
{benavides,jagalindo}@us.es

ABSTRACT

Software product line adoption is a challenging task in software development organisations. There are some reports in the literature of how software product line engineering has been adopted in several companies using different variabil-ity management techniques and patterns. However, to the best of our knowledge, there are no empirical reports on how variability management is handled in companies that do not know about software product line methods and tools. In this paper we present an experience report observing variability management practices in a software development company that was unaware of software product line approaches. We briefly report how variability management is performed in different areas ranging from business architecture to software assets management. From the observation we report some open research opportunities for the future and foster further similar and more structured empirical studies on unaware software product line companies.

KEYWORDS

Experience report, software product lines, unaware variability techniques organisation

1. INTRODUCTION

Software product line engineering is one of the mainstream tendencies in software engineering industrial practices. It is well known that transitioning from a project oriented software engineering method to a software product line engineering approach is not a trivial task [22] and may constitute a strategic decision [3]. There some reports on how companies transitioned towards a product line approach [4, 9, 14, 16, 20, 26]. According some of these reports, for product line adoption success, a strong management commitment is needed as well as a long-term vision. According to [15] smooth transition strategies eliminate many adoption barriers. In this sense, a first step in a smooth transition strategy can be to observe the current product-line-related practices inside an unaware product line organisation, i.e. reporting the existing approaches of the organisation without giving too much details on software product line engineering methods or tools.

To the best of our knowledge, there are no empirical reports of what kind of product line practices are handled in companies that do not know about software product line engineering but that have a positive potential characteristics for the transition. To push forward in this direction, in this paper we present an experience report observing variability management practices in a software development company that was unaware of software product line approaches. One of the authors expended two months working full time inside the company and gave little detailed information about software product line engineering. Methods, processes, tools and other software engineering artefacts were observed and several meetings and interviews were held during that time. After the information gathering, we were able to determine some variability management practices in the company and opportunities for improvement.

We briefly report how variability management is performed in different areas ranging from business architecture to software assets management. We can conclude that although product line engineering concepts were unknown, the company already have some variability management practices developed inside the organisation. These practices were mostly reactive rather than proactive and systematic in the sense that the organisation reacted to reuse and variability management needs after those needs were detected instead of planning for reuse. We also report some open research questions that can be studied in the future and can stimulate further similar and more structured empirical studies on unaware software product line companies because we believe that this can serve as a value input for companies transitioning to a product line and for the research community to discover research opportunities.

1.1 The organisation

The organisation visited in this experience was the "*Servi-*

cio de Rentas Internas" (SRI) [1] in Ecuador, South America. This is the tax agency of the central government of Ecuador. It has more than 3000 workers in all the areas and is divided in eight main national divisions. One of these divisions is the National Division of Technological Development ("*Dirección Nacional de Desarrollo Tecnológico*"), which is responsible for providing software, hardware services and infrastructure for business processes automation as well as citizen and companies relationships with the tax agency. This division has more than 200 employees and is organised in different areas such as Operations, Planning, Development and Networks. At the time of the visit, more than one hundred software products are developed and developed by the company all of them related to the taxes domain.

One of the authors expended two months inside the organisation as a consultant to try to introduce software product line awareness in the institution and improve the development processes inside the organisation. However, the approach was a bit different to other practices to introduce these concepts. The strategy was to observe how variability management is handled in an unaware software product line organisation giving only brief basic concepts to the organisation members. Continuous unstructured and structured meetings and interviews as well as short seminars were developed and the information was later summarised and presented for an agreement among the participants.

The areas studied were related to enterprise architecture and organisation, software artefacts management and development, new project developments, software testing, software development infrastructures and requirements engineering. Also some less time was expended in software maintenance and evolution areas.

Following, we report in different sections the observed variability management techniques in the different areas.

2. VARIABILITY IN ENTERPRISE ARCHITECTURE

Enterprise architecture is the model of all the most important information and behaviour inside a given organisation [12]. There are several enterprise architecture patterns, processes or frameworks such as the Federal Enterprise Architecture [2] or The Open Group Architecture Framework (TOGAF) [12]. In the case of SRI, the reference being used is TOGAF. TOGAF v9 SRI is defines four different enterprise architecture perspectives as shown in Figure 1.

The first one on the top is the Business Architecture (BA) where the most important parts of the organisation are defined such as organisation structure, roles, goals, business services and so forth. The Information Technology Services Architecture (ITSA) is intrinsically related with the BA and should model a product portfolio, interfaces, links between applications with business functions. The Data Architecture (DA) is the one modelling a data catalogue, traceability links between data and business function and services and conceptual models of data. Finally, the Technology Infrastructure Architecture (TIA), which describes a catalogue of technological elements and the relationship among those with the elements of the ITSA.

2.1 Current practices

[1] `www.sri.gob.ec`

[2] `http://www.whitehouse.gov/omb/e-gov/fea`

After some meetings with the members of the team, they concluded that there are points to handle variability in all the four elements of the TOGAF defined architecture. However, there are mainly two points where they considered that variability management should play a key role.

On the one hand, the variability in the BA is key for the organisation. Changes in taxes laws and procedures can change the way all the other architectural components evolve. In this sense the variability in the processes is being handled by means of *business rules* [19]. Business rules can be seen from different perspectives, the business itself and from the IT infrastructure. Business rules oriented approaches try to align business and IT rules. In any case, rules are used as a first-citizen element to model, in this case, variability in the business because one of the ideas is that business rules can change and if they are modelled explicitly the whole systems do not need to change but only the business rules. This approach is useful and it is an improvement with respect to process modelling where no explicit identification of business rules is performed. However, business process variability is still not considered as a whole in the sense that business process families are not identified. Also, although at the process BA level the business rules are modelled, this is not yet completely implemented at the ITSA level which is not an evident transition.

On the other hand, variability management and product line engineering seems to be key in the ITSA especially in the way the product portfolio is exposed to the business. Currently, there are around one hundred different applications that have been developed and are maintained by SRI development unit. According to the information gathered during meetings, there are several groups of applications that could be considered as candidates to make up several product lines related to different domains. This need seems to be a key strategic decision and including in TOGAF a software product line engineering view in the product portfolio level is not explored in the current literature to the best of our knowledge.

Research opportunity : Studying how software product line engineering can complement enterprise architecture models such as TOGAF.

2.2 Potential improvements

A short seminar on configurable business processes was taught to process responsible members of the organisation. A configurable business process [17] is a process that explicitly model variability as a first-class-citizen. It represents a family of similar business processes in a given domain and once configured, it defines a concrete configuration process instance. A simple example is shown in Figure 2. On the left hand size, two different processes are modelled for tax refund. They have some commonality and variability. On the right hand size, a variation point is included and a configurable business process is defined. This is just a naive example of how to model variability in business process families. Previous works proposed to include variability as a modelling element in the business process modelling language [17] or modelling the process variability in a dedicated variability model [8]. Further potential improvements can include the definition of business process families together with business process rules.

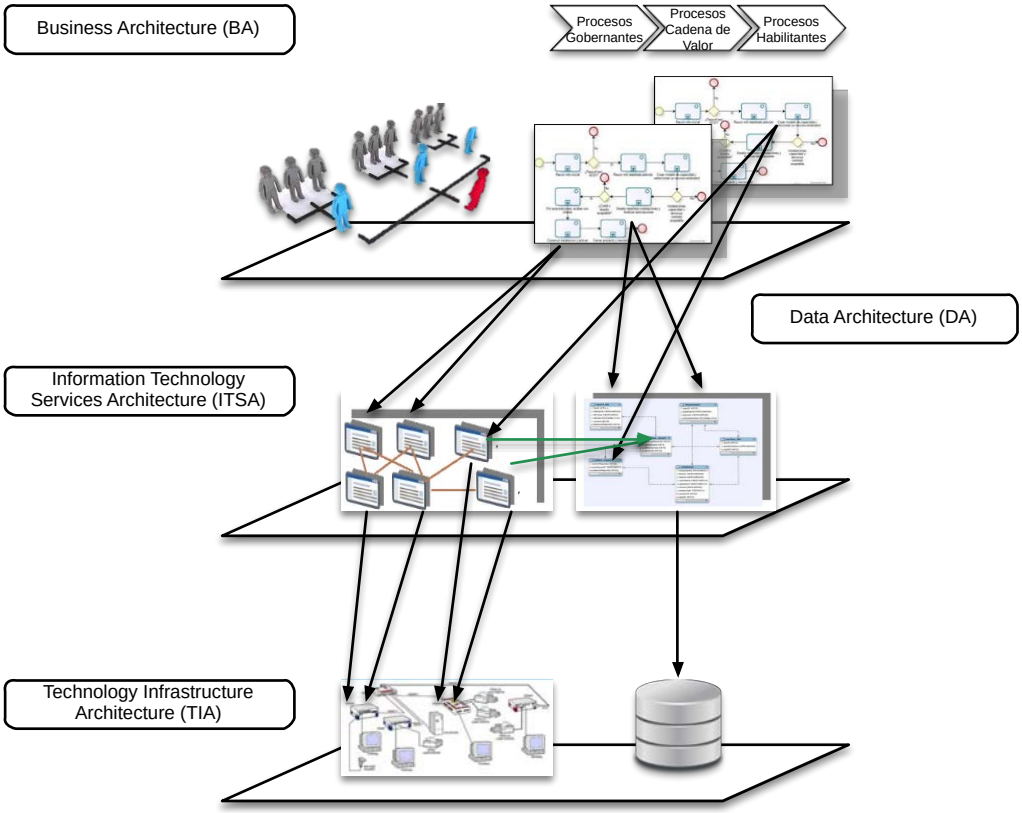

Figure 1: Organisational Architecture at SRI

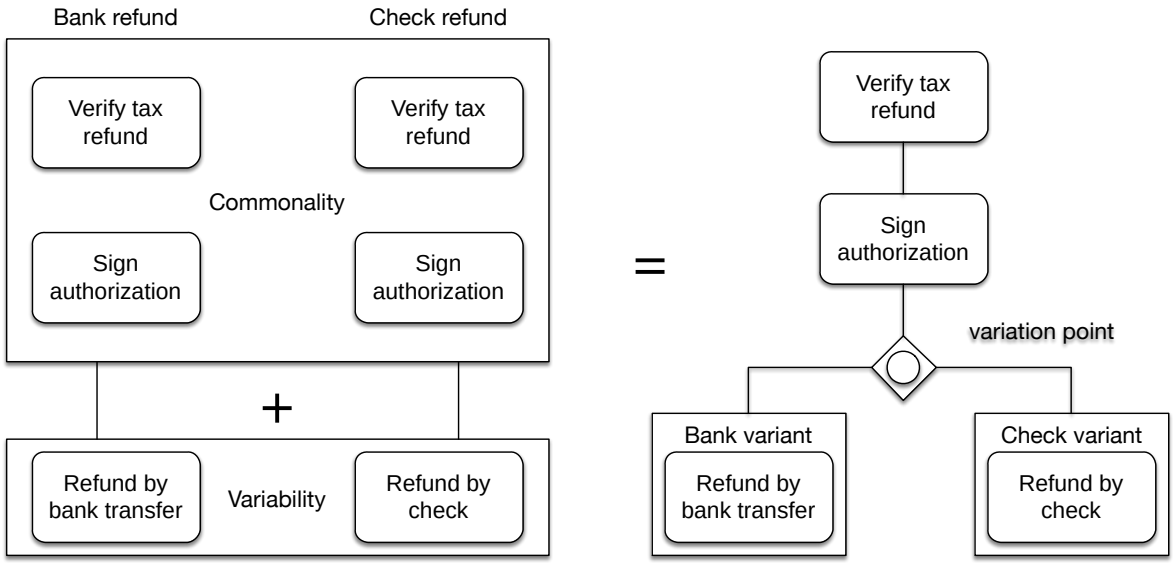

Figure 2: Configurable business process example insipired by examples in [17]

Shifting from a project-centric software engineering model to a software product line engineering model seems to be a key part in the long term at the ITSA level. This software product line strategy shifting is not an evident task and the domain experts reported that this will require a strong team commitment that, with the current size of the institution, can take long time. In this sense, one of the domain experts asked if there are team techniques to foster the variability mentality inside development teams. After an informal literature review in research databases, we did not found such a set of techniques and we envision that this can be an interesting area for future work in the variability management community brining knowledge from cognitive science.

Research opportunity : Studying how variability mentality can be introduced in software developers and managers.

3. ASSETS MANAGEMENT

Although as explained in the previous section there is no software product line strategy defined in the current practices. The development team have gained experience the last decades and detected opportunities for reuse. In software product lines, one of the most important parts of a product line infrastructure is the management of common and variable software assets of a product line.

3.1 Current practices

Currently, there are several software assets that are defined in SRI as "*genéricos*" (generics). Those assets have been built following an opportunistic reuse strategy when similarities in several features of different projects were detected. More than twenty projects from the hundred being maintained and developed at SRI are described as generics. Some of them are key software assets that are used in most of the new and existing projects and some of them are less common assets that are included in some projects while are omitted in some others. In this sense, some software product line *smell* is present in the organisation although it comes from opportunistic reuse needs rather than a well systematic and planned reuse strategy. We can say that there is an incipient division between domain engineering activities (where generics are developed and maintained) and application engineering activities (where generics are tailored and used for product development.

In the organisation, a Continuous Integration (CI) [6] approach together with agile development principles is being introduced in the new projects (currently around six projects are following this strategy). In this sense, a software infrastructure is used including several external applications such as Jenkins [3], Fisheye [4], SVN [5], JIRA [6] or maven [7] among others. The source code repository is organised in several branches and folders according to process and roles needs. Domain experts described this planning as a complicated task since many interactions in the development team as well as in the development process exist.

[3] www.jenkins-ci.org/
[4] www.atlassian.com/software/fisheye/
[5] www.subversion.tigris.org
[6] www.atlassian.com/es/software/jira
[7] maven.apache.org

Inside this software infrastructure SRI uses a repository management tool such as NEXUS [8] to manage internal and external software artefacts. A repository management tool can be roughly defined as a software shelf that allows to handle external and internal software repositories to offer a set of software artefacts to a given organisation. One of the current practices at SRI includes using this tool to include generics artefacts. After an informal literature search we found no experience reports, case studies or any other kind of empirical evidence describing how to use software repository management tools such as NEXUS to handle software assets in a software product line.

Research opportunity : Studying how repository management tools such as NEXUS can be used to manage assets in software product lines.

3.2 Potential improvements

One of the problems described by domain experts is the evolution of generics. It is common to have a generic asset being evolved in parallel in different projects. This later makes it difficult to maintain and merge all the changes made. Concrete guidelines and process have to be defined handling this co-evolution. In this sense, the role of "generic owner/master" was agreed that is needed and will be introduced in the future. This role will be responsible of allowing (or not) changes and commits to a given generic.

Another potential improvement would be to align an eventual software product line strategy with current agile and CI practices in the institution. In this sense some effort in the software product line community have been presented recently [1, 18] but more concrete experiences and work seems to be needed. Especially important is to define and describe how a CI approach can cohabit with a software product line engineering one.

Research opportunity : Studying how continuous integration and delivery practices can be combined with software product line practices.

Generics are used as reusable assets but after some meetings with domain experts they argued that generics can also be seen as product lines because they have variation points and can be tailored to work in different domains. In this sense, a multiple product line vision can also be considered in the future [11, 5].

4. COMMON BASE ARCHITECTURE

At SRI, a good amount of projects are web-based projects. Those projects have been developed in the last decade or so. After some time, the development team detected that most of the projects had a common base architecture.

4.1 Current practices

Currently, the new projects are developed from what the team call the "base project". This project is a sort of skeleton (a.k.a. archetype) for starting any new web development project and includes a common architecture as well as commonly used configuration files and features. The base project is tested and certified by the Q&A unit and it is indexed in the repository manager (NEXUS) and can be retrieved with the build management tool (maven). Once

[8] www.sonatype.org/nexus/

installed, the development team can remove parts as needed depending on the kind of project or task. Here we detect what is known as *negative variability*, i.e. the base project has internal variability because there are common parts and optional ones but all the optional parts are included in the project and then the development team has to remove the optional parts that are not required.

4.2 Potential improvements

One of the potential improvements would be to have in the base project a configuration process to include the generics explained in Section 3. According to domain experts, it would be of great help to have a sort of wizard or configurator that allowed to select or deselect artefacts to be included in the base project following a *positive variability* approach in the sense that required optional features are added to the base project according to specific needs. A feature model base configurator [2] can eventually help in this task.

The possible evolution of the base project is a challenging task. It might happen that parallel projects use the base project at the same time but some of them make changes in the project because some bugs are detected or some features are added. Also, technology evolution can make the base project obsolete in some scenarios (e.g. evolution of the Java virtual machine or SDK version used). Although at the moment the base project has not evolved, it is recognised by the domain experts that this will happen in the future. Therefore, techniques and processes have to be described to handle this evolution. The management of the evolution of generics is mainly done manually and automated mechanisms are expressed to be desirable.

Research opportunity : Studying automated mechanisms and processes for software product line and assets evolution.

5. DEPENDENCY MANAGEMENT AMONG ASSETS

Dependency management among assets is a crucial task in software product line engineering. It has to do with the dependency, relationship and traceability that exist among the different software assets and how to handle these dependencies. There are different levels of dependency management going from high-level features to features representing fragments of code.

5.1 Current practices

At the time of the visit, SRI is following TOGAF v9 recommendations. Therefore, a map of relationships among the different projects is being established and documented. With the help of domain experts a catalogue of applications together with the dependencies and relationships is being documented. In this sense, some not well-defined domain engineering activities are being performed like domain scoping and analysis but not structured or following concrete domain engineering guidelines. One of the set of projects being documented and one of the most important ones in this task according to domain experts is the so-called generics (see Section 3). In this sense, a high level dependency map of these assets is being constructed and a variability model was saw as a good mechanism to model these artefacts. However, it was difficult, if not impossible, to construct a hierarchically arranged feature model because there was not a hierarchical structure in the relationships of generics. Instead, a spreadsheet is being used.

At lower level, dependencies are also modelled and managed using the maven capabilities. Maven [21] is a project management tool at development level. It allows automated building as well as testing, deployment and so forth using different plug-ins and complementary tools covering most of the development life cycle. Maven, as a dependency mechanism worth being analysed from a variability management perspective. Maven configuration files contain advanced dependency management capabilities that are widely exploited in industry. At SRI, maven is used for the project management life-cycle and the dependencies among assets are described in the maven configuration files. It is possible to say that a maven file is able to describe a variability model similarly to other low-level languages such as the Debian package description files [7].

5.2 Potential improvements

It might be interesting to use more structured domain analysis techniques to better scope the domain and retrieve the dependency map and variability model at a high level. Also, the traceability among the high-level dependency management described in the TOGAF architecture and the more technical and detailed dependencies expressed with maven are far from being automated. It may be of help to have automated mechanisms and tools to trace variability models to build and project management scripts like maven. In this sense, the work in [10] may help.

Research opportunity : Studying how feature models or any variability model can trace to build management tools such as maven, ant or Make.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have presented an experience report on how variability management techniques are used in a company that was not aware of software product line concepts. We detected that variability management is performed all along the organisation structure and that is seen as a need in most of the areas studied. From the observation we detected and reported research opportunities. Moreover, we paved the way for further structured empirical research in unaware software product line companies to learn what they do in practice and how they can improve getting insights of new research opportunities.

However, there is still work to do in the research presented in this paper:

- **Methodology.** Improve the description of the research project will help to enhance the probabilities of succeeding when introducing product lines methodology in the organisation as proposed in the literature [24].
- **Learn from the past.** Previous research pointed out some problems when transitioning to a variability-aware development. For example, Staples et al.[25] showed how this process was undertook in the HP Company. Moreover, literature not directly related to software product line adoption [23, 13] can inspire this research for a better approximation to the existing problem in the company.

Acknowledgments

We would like to thank all the members of the SRI that participated in the meetings, interviews and discussions. This work has been partially supported by the Prometeo programme by SENESCYT, Ecuador, by the European Commission (FEDER), the Spanish Government under project TAPAS (TIN2012-32273) project and the Andalusian Government under projects THEOS (TIC-5906) and COPAS (P12-TIC-1867).

7. REFERENCES

[1] Muhammad Ali Babar, Tuomas Ihme, and Minna Pikkarainen. An industrial case of exploiting product line architectures in agile software development. In *Proceedings of the 13th International Software Product Line Conference*, pages 171–179. Carnegie Mellon University, 2009.

[2] David Benavides, Alexander Felfernig, José A Galindo, and Florian Reinfrank. Automated analysis in feature modelling and product configuration. In *Safe and Secure Software Reuse*, pages 160–175. Springer, 2013.

[3] Gunter Bockle, Paul Clements, John D McGregor, Dirk Muthig, and Klaus Schmid. Calculating ROI for software product lines. *Software, IEEE*, 21(3):23–31, 2004.

[4] Paul C Clements, Lawrence G Jones, John D McGregor, and Linda M Northrop. Getting there from here: a roadmap for software product line adoption. *Communications of the ACM*, 49(12):33–36, 2006.

[5] Deepak Dhungana, Dominik Seichter, Goetz Botterweck, Rick Rabiser, Paul Grunbacher, David Benavides, and Jose A Galindo. Configuration of multi product lines by bridging heterogeneous variability modeling approaches. In *Software Product Line Conference (SPLC), 2011 15th International*, pages 120–129. IEEE, 2011.

[6] Martin Fowler and Matthew Foemmel. Continuous integration. *Thought-Works www.thoughtworks.com/ContinuousIntegration.pdf*, 2006.

[7] José Galindo, David Benavides, and Sergio Segura. Debian packages repositories as software product line models. towards automated analysis. In *ACoTA*, pages 29–34, 2010.

[8] Gerd Gröner, Marko Bošković, Fernando Silva Parreiras, and Dragan Gašević. Modeling and validation of business process families. *Information Systems*, 38(5):709–726, 2013.

[9] William A Hetrick, Charles W Krueger, and Joseph G Moore. Incremental return on incremental investment: Engenio's transition to software product line practice. In *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, pages 798–804. ACM, 2006.

[10] Gerald Holl, Christoph Elsner, Paul Grünbacher, and Michael Vierhauser. An infrastructure for the life cycle management of multi product lines. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 1742–1749. ACM, 2013.

[11] Gerald Holl, Paul Grünbacher, and Rick Rabiser. A systematic review and an expert survey on capabilities supporting multi product lines. *Information and Software Technology*, 54(8):828–852, 2012.

[12] Andrew Josey. *TOGAF Version 9: A Pocket Guide*. Van Haren Publishing, 2009.

[13] Daniel Karlström. Introducing extreme programming-an experience report. In *Proceedings of the 3rd International Conference on eXtreme Processing and Agile Processing Software Engineering (XP2002)*, 2002.

[14] Charles W Krueger. Easing the transition to software mass customization. In *Software Product-Family Engineering*, pages 282–293. Springer, 2002.

[15] Charles W Krueger. New methods in software product line practice. *Communications of the ACM*, 49(12):37–40, 2006.

[16] Pasi Kuvaja, Jouni Similä, and Hanna Hanhela. Software product line adoption–guidelines from a case study. In *Software Engineering Techniques*, pages 143–157. Springer, 2011.

[17] Marcello La Rosa, Marlon Dumas, Arthur HM Ter Hofstede, and Jan Mendling. Configurable multi-perspective business process models. *Information Systems*, 36(2):313–340, 2011.

[18] Kannan Mohan, Balasubramaniam Ramesh, and Vijayan Sugumaran. Integrating software product line engineering and agile development. *Software, IEEE*, 27(3):48–55, 2010.

[19] Tony Morgan. *Business rules and information systems: aligning IT with business goals*. Addison-Wesley Professional, 2002.

[20] Dirk Muthig. *A Light-weight Approach Facilitating an Evolutionary Transition Towards Software Product Lines*. PhD thesis, University of Kaiserslautern, 2002.

[21] Timothy M O'Brien. *Maven: The Definitive Guide*. O'Reilly, 2008.

[22] Klaus Pohl, Günter Böckle, and Frank Van Der Linden. *Software product line engineering: foundations, principles, and techniques*. Springer, 2005.

[23] Susan Rosenbaum and Bertrand du Castel. Managing software reuse—an experience report. In *Proceedings of the 17th international conference on Software engineering*, pages 105–111. ACM, 1995.

[24] Per Runeson, Martin Host, Austen Rainer, and Bjorn Regnell. *Case study research in software engineering: Guidelines and examples*. Wiley. com, 2012.

[25] Mark Staples and Derrick Hill. Experiences adopting software product line development without a product line architecture. In *Software Engineering Conference, 2004. 11th Asia-Pacific*, pages 176–183. IEEE, 2004.

[26] Stefan Strobl, Mario Bernhart, and Thomas Grechenig. An experience report on the incremental adoption and evolution of an spl in ehealth. In *Proceedings of the 2010 ICSE Workshop on Product Line Approaches in Software Engineering*, pages 16–23. ACM, 2010.