Product Line Engineering for NGO Projects

Fabio Calefato[*], Roberto De Nicolò[†], Filippo Lanubile[*], Fabrizio Lippolis[†]
[*]University of Bari, Italy
Email: {fabio.calefato, filippo.lanubile}@uniba.it
[†]Informatici Senza Frontiere, Italy
Email: {roberto.denicolo, fabrizio.lippolis}@informaticisenzafrontiere.org

***Abstract*—Non-governmental organizations (NGOs) are often plagued by very limited human and financial resources. In this paper, we show how product line engineering (PLE) offers an opportunity to increase the sustainability of software projects that rely on the help of NGO volunteers. Building on the case of an Italian NGO that supports assistive technologies, we propose a PLE model that only depends on the branching capability of a free version control system.**

***Index Terms*—Assistive technologies, NGO, sustainable software development, product line engineering, branching.**

I. Introduction

IT without borders (shortly ISF, from the Italian name "Informatici Senza Frontiere") is an Italian non-governmental organization (NGO) that considers information technology (IT) as an asset of primary necessity and then an essential prerequisite for the economic and social development. Founded in 2005 by a small group of Italian IT professionals, ISF today counts 13 regional sections and over 300 members. ISF organizes its volunteers to work on computing projects, focusing its action on contexts of marginalization, difficulties and emergencies, both in Italy and in developing countries. ISF projects tend to involve other NGOs and include different stakeholders, ranging from schools and hospitals to marginalized communities, disabled people and senior citizens.

Since it is primarily run by volunteers offering their spare time, ISF has very limited human and financial resources. The University of Bari is helping ISF to sustain the evolution of three major software projects for assistive technology, by recruiting, as developers, student volunteers working at their final-year theses. Yet, the issues coming from high turnover rate and the limited amount of time that volunteers have to work on projects still remain.

In order to counteract such problems and then further increase project sustainability, we propose a product line engineering (PLE) that only depends on the branching capability of a free version control system.

II. NGO Projects

In this section, we illustrate three of the most successful prototypes of assistive technology developed through the NGO-academic partnership.

A. Paperboy

Paperboy (aka Strillone, in Italian) allows visually impaired people to listen to Internet newsfeeds on demand by leveraging vocal synthesis. Paperboy was developed upon request from the Italian Blind Union (IBU) association who reach out to ISF asking for help to improve the accessibility of the local newspaper for their own members.

Paperboy was initially developed as a web application for PCs, compatible with most of the screen readers for the visually impaired available on the market. Then, as blind people are avid smartphone users, we decided to provide them with mobile solutions too, developing apps for Android, iOS and Windows Phone.

Since its release on all the three app stores, Paperboy has been received very well. In fact, as of this writing, it has been already downloaded over 4,000 times overall. Finally, Paperboy has been recently nominated finalist in the "Access to information and knowledge" track of the *2014 WSIS Prize*[1], a contest organized by the International Telecommunication Union (ITU, the United Nations specialized agency for information and communication technologies) which awards the best IT projects in the world [1].

B. I Speak Again

I Speak Again (ISA) is a communicator designed for people who are anarthric and quadriplegic, and then unable to speak and move their limbs. The goal of ISA is to bring an affordable technological solution to people who cannot communicate because of illness.

Back in 2011, ISF received a request from an amyotrophic lateral sclerosis (ALS) association to help patients to communicate with friends and relatives. People affected by ALS use very expensive eye-tracking devices that allow them to compose words on a screen and play them back through vocal synthesis. Communicators like these are so expensive (about 20.000€) that the Italian public health service is unable to provide them to all patients in need.

ISA can be described as a virtual, on-screen keyboard controlled through the eye movement, using any device equipped with an eye-tracking module and capable of running a web browser. Other than text to speech synthesis, a notable

[1] http://groups.itu.int/stocktaking/WSISProjectPrizes2014.aspx

feature is the use of a text predicting and correcting algorithm to expedite sentence composition in both English and Italian. Besides, ISA allows selecting built-in images and common sentences that express primal needs like "*I feel cold*", "*I want to eat*" or "*I'm tired*". Except for a commercial eye-tracking module, to keep overall costs as low as possible, we built ISA using exclusively open source components.

C. I Move Again

Other than being unable to speak, ALS patients cannot move either. Therefore, after completing a prototype that can help them to "speak again", the natural next step was helping them to "move again". This is how the *I Move Again* (IMA) project started as a spinoff of ISA. In fact, we envisioned that the gaze input could be used to control the movement of a power wheelchair.

On the software side, we decided to leverage the infrastructure already available in ISA, especially as far as presentation and eye tracking is concerned. According to our design approach, the screen has been split into five main areas, corresponding to the four directions (i.e., *left*, *right*, *forward*, and *back*) plus the *stop* command.

As for the hardware, the eye tracking movement captured by ISA module is sent over network to an Arduino Uno board mounted on the wheelchair. Connected to this board is an Arduino-programmed relay board that actually transforms the eye-gaze input into signals for the power wheelchair engine. In other words, when gaze is directed onto, say, the back area of the screen, the boards mounted on the wheelchair activate one relay simulating the input that the joystick would provide if it were moved backward.

III. The Case for Product Line Engineering

A software product line aims at increasing quality and development speed while decreasing costs. For example, large companies such as Nokia and HP experienced large increases in productivity since turning to software product lines [1]. Such benefits, however, are not limited to large companies only. Several are the reports of product lines being beneficial to small-medium companies as well [4][5][13][14]. Besides, software product lines incorporate various practices that have proved to be successful in managing distributed open source software projects [10][11][12]. It has also been argued that the Linux Kernel represents the best example of a large open-source product line, considering how the community successfully manages the development of OS kernels for very many and different software architectures [6].

The NGO projects of assistive technology are growing as families of related systems, which vary according to the type and degree of disability. We then propose to evolve them by reusing a platform of common aspects (i.e., core assets) while unique features are allowed to differ. The benefit of software product lines that is more relevant for the NGO project is the decrease in labor needs [7][9].

Large companies usually rely on commercial solutions like Gears to support the product line lifecycle. However, the choice of the infrastructure upon to build a product line-compliant process for a NGO, is constrained by limited or no budget for acquiring commercial software solutions. Our proposal is to accomplish the benefits of tool support for product lines, without changing the existing communication and development infrastructure at the NGO. In our case, the development activities are coordinated through GitHub and email. This is not so different from software engineering practices in the Linux community, which builds its product line of kernels using a lean infrastructure that ultimately relies on computer-mediated communication tools (e.g., mailing lists and chats), web-based knowledge centers (i.e., website, wiki, blog), and version control systems (i.e., git).

Software product lines are normally built by first developing the platform of core assets and then adding incrementally products as well as new core assets are needed [2][8]. Yet, sometimes software product lines are built using a *reactive approach*, that is, starting with one or more existing products from which the core assets are generated for future products [1]. As compared to the other one, the retroactive approach grants lower entry costs.

We are planning to follow this retroactive approach to start a software product line from the existing NGO products. Of course, we will need to redefine the common architecture and the core assets in a way that they are enough robust, extensible, and appropriate to future product line needs. Table 1 shows the core assets and the variabilities for the product line that we are going to build from ISA and IMA products. In particular, the new, refactored ISA is a product that is tailored for people suffering for speech impairment, whereas IMA is intended for those affected by motor impairment.

TABLE I. Commonalities and Variabilities of The Product Line for ISA and IMA

	Features	**ISA**	**IMA**
Core assets	Eye tracking	✓	✓
	UI dashboard	✓	✓
	DB access	✓	✓
	Networking	✓	✓
Variabilities	Virtual keyboard	✓	
	Messaging	✓	
	Power wheelchair control		✓
	Home automation		✓

When installed together, ISA and IMA will provide a comprehensive solution for people affected by totally invalidating diseases such as ALS.

Afterwards, we will create a product line for the Paperboy apps too. In this case, however, the product line is not intended to manage feature variabilities across products (apps are written in different languages), but rather to handle differences within each mobile platform (e.g., different screen sizes, API changes across OS versions).

IV. BRANCHING MODEL

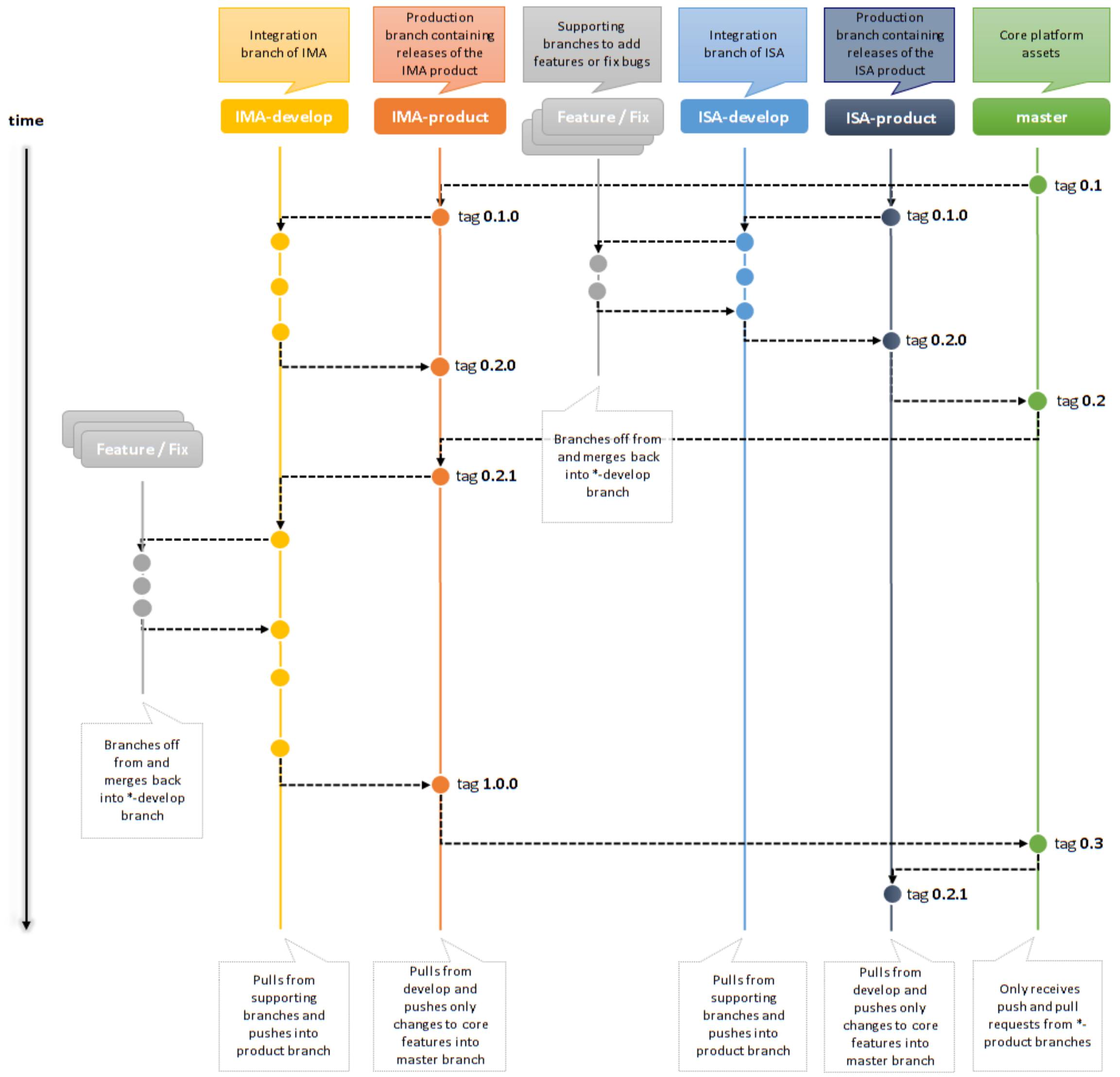
Being the existing infrastructure for software development, the NGO product lines will rely on the git version control system (VCS) and the GitHub project-hosting repository. In Fig. 1, we illustrate a graphical representation of the branching model that we are adopting at ISF to ease the release management of the NGO product families. For the sake of readability, the model is instantiated for the specific case of the ISA/IMA product family. The branching model, which is an adaptation of the one proposed by Driessen [3], meets the requirement of only needing a distributed VCS, such as git in our case.

First, we consider the git repository hosted at GitHub (the *origin* in git lingo) to be the "central" one. Technically, there is no such a thing as a centralized repository in a distributed VCS. Yet, all the NGO developers, who must fork the origin into a personal clone of the repository, are invited not to push into or pull changes from other peers' repositories. Instead, all changes must propagate through the origin. In this way, we call "origin" the central repository too. Such a restriction is particularly helpful to the NGO members in charge of the project because it ensures less coordination effort in tracking pending contributions.

In our model, we distinguish between *main branches* and *supporting branches.* Main branches exist for the entire life of the project. In particular, the central repository holds *master*, **-product*, and **-develop* branches, where the last two are instantiated for each product – in our case, ISA-product, IMA-product and ISA-develop and IMA-develop.

In the origin, the master branch, which could be renamed to *core*, is where the source code of the shared assets of the products is committed. In our case, the common features shared between ISA and IMA include, for example, the eye-tracking module, the UI dashboard, and the data access layer. Each commit made to the origin/master branch must be tagged with a release number (e.g., 0.1).

Instead, the features that set the products apart must go into the product-specific branches. In other words, the origin/*-



Integration branch of IMA
Production branch containing releases of the IMA product
Supporting branches to add features or fix bugs
Integration branch of ISA
Production branch containing releases of the ISA product
Core platform assets
time
IMA-develop
IMA-product
Feature / Fix
ISA-develop
ISA-product
master
tag 0.1
tag 0.1.0
tag 0.1.0
tag 0.2.0
tag 0.2.0
tag 0.2
Feature / Fix
Branches off from and merges back into *-develop branch
tag 0.2.1
Branches off from and merges back into *-develop branch
tag 1.0.0
tag 0.3
tag 0.2.1
Pulls from supporting branches and pushes into product branch
Pulls from develop and pushes only changes to core features into master branch
Pulls from supporting branches and pushes into product branch
Pulls from develop and pushes only changes to core features into master branch
Only receives push and pull requests from *-product branches

Fig. 1. The ISF branching model for handling release management of ISA/IMA products (adapted from [3]).

products are the *production branches* where the source code always reflects a production-ready state. In our example, chat and virtual keyboard features belong to the ISA-product branch, whereas home automation and wheelchair control belong to IMA-product. Every commit to production branches creates a new release, which therefore is tagged to reflect a version increment, such as, from 0.1.0 to 0.2.0. The origin/master branch receives push commits and pull requests from production branches. Push commits are necessary to propagate changes made to the common features from one product to the others in the family. Performing a pull request from the master branch into a production branch also creates a new product release that is tagged to reflect a minor increment, e.g., from version 0.2.0 to 0.2.1.

The origin/*-development branches, instead, are the so-called *integration branches*. Here the source code contains the latest delivered development changes for the upcoming new release. If we had continuous integration, nightly builds of products would be automatically compiled out of these branches. Changes from integration branches are merged into their respective product branches when a development release is ready for production.

The other types of branches are called *supporting*. These branches will not be found in the central repository, but only in its clones. In fact, these branches are typically created in developers' forks just to implement new features or fix bugs. As such, they have a limited lifespan, considerably shorter than that of the project. They will only exist as long as the feature is in development or the bug fixed. Therefore, they will be eventually merged back into the development branch they were branched off from and then deleted.

V. Conclusions

In this paper, we have presented the case of a NGO whose volunteers develop assistive technologies without any financial support. We have argued that PLE may offer an opportunity to increase the sustainability of NGO projects and have proposed a PLE model that only relies on the branching capability of a free version control system.

When implemented, we expect the proposed PLE model to bring benefits in terms of decreased labor needs and time required to deliver fixes and new features. This is fundamental for the NGO since its projects are carried on through the commitment of its volunteers in their limited free time. We also expect that the proposed PLE model will make easier project release management

Acknowledgment

We would like to thank the head of the Apulian ISF chapter and all the students who contributed to ISF projects during their final-year thesis projects.

References

[1] L. Baas, P.C. Clements, and R. Kazman. Software Architecture in Practice, 2nd ed. SEI Series in Software Engineering. Addison-Wesley, 2003.

[2] F.J. Linden, K. Schmid, E. Rommes, Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering Springer, 2007.

[3] V. Driessen, "A successful Git branching model", http://nvie.com/posts/a-successful-git-branching-model.

[4] C. Gacek, P. Knauber, K. Schmid, & P. Clements, "Successful Software Product Line Development in a Small Organization. A Case Study". Technical Report, Fraunhofer Institut for Experimental Software Engineering (IESE), 013.01/E, 2001.

[5] P. Knauber, D. Muthig, K. Schmid, and T. Widen. "Applying Product Line Concepts in Small and Medium-Sized Companies." IEEE Software, vol. 17, no. 5, Sept. 2000, pp. 88-95, DOI=10.1109/52.877873.

[6] R. Lotufo, S. She, T. Berger, K. Czarnecki, and A. Wąsowski, "Evolution of the Linux Kernel Variability Model." Software Product Lines: Going Beyond, Lecture Notes in Computer Science, vol. 6287, 2010, pp 136-150, DOI=10.1007/978-3-642-15579-6_10.

[7] L.M. Northrop, "SEI's Software Product Line Tenets." IEEE Software, vol. 19, no. 4, Jul. 2002, pp. 32-40, DOI=10.1109/MS.2002.1020285.

[8] K. Pohl, G. Böckle, F.J. van der Linden, Software Product Line Engineering: Foundations, Principles and Techniques, Springer, 2005.

[9] SEI CMU, "Software Product Lines case studies" http://www.sei.cmu.edu/productlines/casestudies.

[10] K. Tate, Sustainable Software Development: An Agile Perspective, Addison-Wesley, 2005.

[11] F. van der Linden, "Open Source Practices in Software Product Line Engineering." In Software Engineering, Lecture Notes in Computer Science, vol. 7171, 2013, pp 216-235, DOI=10.1007/978-3-642-36054-1_8.

[12] J. van Gurp, C. Prehofer, and J. Bosch, "Comparing practices for reuse in integration-oriented software product lines and large open source software projects." Softw. Pract. Exper, vol. 40, no. 4, Apr. 2010, pp. 285-312, DOI=10.1002/spe.v40:4.

[13] M. Verlage and T Kiesgen, "Five years of product line engineering in a small company." In Proc. 27th Int'l Conf. on Softw. Eng. (ICSE '05), St. Louis, USA, May 15-21, 2005, pp. 534-543. DOI=10.1145/1062455.1062551.

[14] T. Vernazza, P. Galfione, A. Valerio, G. Succi, P. Predonzani, "Moving Towards Software Product Lines in a Small Software Firm: a Case Study." In Proc. of ICSE Workshop on Software Product Lines: Economics, Architectures and Implications, Limerick, Ireland, June 2000.