Facts, trends and challenges in modern software development

E. Michael Maximilien*

IBM Research,
650 Harry Road, E3-418,
San Jose, CA 95120, USA
E-mail: maxim@us.ibm.com
*Corresponding author

Pedro Campos

Madeira-ITI,
University of Madeira,
Campus Universitário da Penteada,
9000-390 Funchal, Portugal
and
INESC-ID,
Rua Alves Redol 9,
1000-029 Lisboa, Portugal
E-mail: pedro.campos@inesc-id.pt

Abstract: The IT industry is not new to change and evolution, however, we are now in an era of two fundamental waves of IT changes. First, the post-PC era, where mobile devices and tablet-like devices are giving end-users the ability to consume information when they want it and where they want it. Second, the post-server era where companies no longer need to neither buy nor provision servers in their own data centres but instead rent the compute resources as needed. This twin change has direct consequences to how end-users consume software, how that software is produced, and how it is delivered.

Keywords: agile method; trends; software engineering.

Reference to this paper should be made as follows: Maximilien, E.M. and Campos, P. (2012) 'Facts, trends and challenges in modern software development', *Int. J. Agile and Extreme Software Development*, Vol. 1, No. 1, pp.1–5.

Biographical notes: E. Michael Maximilien is a Research Staff Member at IBM Research. Prior to joining IBM Research's division, he spent ten years at IBM's Research Triangle Park, NC, in software development and architecture. He led various small- to medium-sized teams, designing and developing enterprise and embedded Java software; he is a founding member to three worldwide Java and UML industry standards. His primary research interests lie in distributed systems and software engineering for the web; in particular, web APIs, mashups, cloud computing, social software, and agile methods and practices. Reach him via his website (http://www.maximilien.com) and Twitter@maximilien.

Pedro Campos is an Assistant Professor at the University of Madeira, Portugal, and an Invited Researcher at the Visualisation and Intelligent Multimodal Interfaces Group, INESC ID Lisbon. He is also the Program Director for the BSc in Computer Science and a Founding Member of the IFIP 13.6 Working Group on Human Work Interaction Design. His main research interests are agile software development, UML, tool support, interaction design and human-computer interaction.

1 Introduction

We are in the midst of two important changes in the IT industry that are causing ripple effects in all aspects of software development, delivery, and consumption. These two changes are typically noted as 'entering the post-PC' and 'entering the post-server' or cloud computing era.

The post-PC era is manifest with a move toward increasing usage of mobile computing devices such as cell phones, tablets, and other portable devices. These mobile computers have the same compute capacity as entry-level laptops but have some constraints while also offering new features. The key constraints are:

a smaller screen for output

b touch or voice interface as input.

The principal new features that these devices offer, besides portability, are constant connection to the internet and the fact they are also communication devices with phone-like capabilities.

The post-server era is manifest with a move by companies (small and large) to use compute resources that are not on-premise but rather accessed via the internet or 'the cloud'. This means that no longer are companies' IT department building dedicated data centres or provisioning departmental servers, they are rather renting the needed compute capacity from cloud computing providers. The advantages are lower costs and only paying for what is needed versus paying up-front capital costs and risking under- or over-provisioning compute resources. Cloud computing providers use their expertise and hardware economies of scale to offer the compute resources at very low price points – in the order of 10 cents per hour.

These two trends are changing the behaviours and workflows of consumers and enterprise IT departments. However, do they also have other impacts to the software industry? In particular, what is the impact to software lifecycle? How does a post-PC and postserver era change the way software is designed and developed? Also, could tailored versions of existing development processes be more appropriate for cloud development?

2 Mobile computing trend

The mobile computing trend has three main realities that affect usages and the types of software created for these devices. First, devices have limited screen real-estate. This means that the output for these devices is constrained and must be economised. With

recent phones the entire device constitutes the screen, but they are still limited to a form factor that can fit easily in a shirt pocket. The implication for software development is primarily in human factor engineering and understanding the many new contexts that these devices are being used.

With mobility comes constant connectivity. These devices are often communication devices (cell phones) but are also increasingly larger consumption devices with some connection to the internet. The combination of mobility and connectivity means that location-based services and applications are finally a reality. With a user's permissions the application can infer the location of the user and offer capabilities and features that are tailored to this newfound contextual information. New devices like the iPad or the Android tablets allow developers means to exploit the potential beneath this mobility and connectivity mixture.

The final reality of the mobile computing trend that we would like highlight is the new types of input mechanisms for these devices. There are primarily two types of input: touch and voice. These two modes of input are more natural for mobile devices since by virtue of being used on-the-go, keyboards and mice are unlikely to be attached to these devices. Touch is a rich and natural user input while voice allows for an input means that requires less visual focus from the user. Therefore, we achieve the ability to perform operations as the user is performing other tasks, e.g., driving, walking, and so on.

As a consequence to the mobile computing boom, software applications are built to have not only newer (more purposeful) interfaces, with a variety of modern input modes, but have also become smaller and more specialised. The '90s trend of creating suites of software that included a myriad of features and sold for very high premium is now completely reversed. Applications for mobile computing devices are small and specialised and sell for very low costs and sometimes are completely free. Mobile applications are complemented with web services and integrate with each other to extend features, e.g., social collaboration via Twitter and Facebook.

The challenges in software development come first in the fact that these mobile applications are built quickly with intent to fill a specialised and a niche space. Since the applications sell for low costs, the developers are looking to recoup development costs with sales volumes as well as cross-promotions and cross-selling of advertisement or of additional features that extend the application. The impact on software development processes is to push the agile techniques to their limits as well as questioning some long held tenets such as involving the customer and test-driven development.

Involving a customer that is mobile has its own set of challenges; these can be overcome but to do so, one needs to primarily try to replicate the contexts in which the customer intends to use the software. Such context replication is especially difficult when considering that the connectivity and ambient location of the customer significantly impact their usage experiences... Further, since the application is designed for a mobile device, many of the assumptions that one had when developing applications for desktops or laptops are no longer valid. The end user in question is likely involved in some primary activity and using the application to complement that primary activity or to kick start a secondary activity. Understanding these user contexts and designing applications for them is a key. Testing is increasingly difficult, since the contextual changes are so varied and frequent that they are hard to fully replicate.

3 Cloud computing trend

The second trend is the movement toward renting computing resources on the internet. It is well-documented that cloud computing providers can offer these compute resources at prices that are difficult to match if one was to create their own data centres. Additionally, since users can scale up and down based on their needs, the economics of cloud computing are difficult to match otherwise. This is the primary result of efficient allocation of compute resources to those who need it the most at the time they need it.

However, it is worth noting that while cloud computing impacts the economics of software development and deployment, it is also affecting how software is designed built and maintained. First, since cloud computing servers are rented as needed, the cost of pushing computation to servers has never been more affordable. This means that the application architectures designed for the cloud push data and computation to servers. This complements really well the trend to mobile devices, which have limited screen and capacity and thus, are perfectly suited to act like view ports to cloud services.

Other aspects of cloud-based software development have to do with testing and experimentation. First, since server costs are marginal, continuous integration and continuous testing is more of an assumed reality. Additionally, the ability to experiment with high-computational loads is more affordable. Features that required integrating with large databases or features that required heavy computation per user are now possible... Scaling is not a major concern, since more computing resources can be thrown to the problems for relatively low costs. Cloud platforms are also making the scaling concerns disappear by promising automatic scaling as needed and in a more principled fashion.

Naturally, cloud computing also comes with its own set of challenges that also impact software development. Principally, these come in terms of governance and management of applications in the cloud. Since the application's data resides in remote servers, caution must be taken on how secure this data is and the associated privacy concerns for sensitive data. Governing access to the cloud resources and managing how software versions are pushed and rolled back becomes a more serious concern since as these development and maintenance tasks are easily performed and therefore are also more likely abused.

4 Collaborative development

These two new trends (mobile and cloud) raise some interesting questions about how software development processes can be improved and can help developers create better software. One such approach is via collaborative development. Extreme version of collaborative development is well known and used in open-source development (OSS), which has increasingly produced mission-critical software and some of the most used software today, e.g., Linux and Apache server.

However, collaborative development can also be done in smaller scale and have more social characteristics. Services such as GitHub, Google Code, and Jazz.net are great examples of collaborative development tooling and services that accommodate small to large teams; and provide features that can help the development process be more dynamic and is done completely on the cloud.

Additionally, to address some of the fast testing cycles for this new class of developed software, some companies, such as Facebook, are increasingly making use of

their users for beta testing and for providing bug report on new features. This type of crowdsourcing of quality assurance (QA) is not without its risks, however, it enables the participation of the most active users and allows the software development, deployment, and production cycles to be practically immediate.

Naturally, using a more collaborative development process poses its own set of challenges. First, there tends to be many orphan projects in the OSS ecosystems. Lots of projects are started but fail to materialise into something concrete and useful. This is also the consequence of the real challenges to create and nurture developer communities. Without the right incentive structures, collaborative development could be more of a challenge than a solution.

5 This issue

In this first issue of the *International Journal of Agile and Extreme Software Development*, we look at real world case studies covering some of the trends and challenges we have outlined.

First, Mangudo et al. present a case study on how a critical business process application development was saved by using the crystal clear methodology. The authors discuss the evolution of the project from desktop to the web and how the agile method help solve some of their teaming issues.

Second, Hoda et al. delve into the often asked question of how much documentation is needed for a software team using agile methodologies? The typical answer is doing just enough documentation. Of course, such an answer is not precise and difficult to measure. Using a study of 58 teams and 23 organisations using various methodologies, they catalogue the different documentation strategies for development, testing, and maintenance. In doing so, they also devise a means for knowing how to measure what is 'good enough' documentation.

Fitting the challenges outlined in the trend to new style of computing, such as mobile, Campos' paper shows how agile techniques can be used for various types of 'unusual' devices with new styles of interactions. These go beyond touch interfaces for phone-like devices but also for interactive devices in stores and in other social settings.

Barksdale and McCricard look into how software innovation in an agile context can be done by engaging usability teams. They propose a research methodology along with an agenda that can be used by agile teams to result in more innovative software teams. One of their key insights is to expand software development teams to multidisciplinary social interactions, especially in the context of usability. They also look at the consequences of not using a socially integrated approach to usability in software development and its consequences.

Finally, Sarja does a first of a kind overview study of a star start-up company: 37 signals, which is often cited as an exemplar model for newer cloud-based software development. Sarja reviews the 'getting real' principles that the founders of 37 signals have laid out as important to their start-up culture and software development processes. In doing so, Sarja tries to also compare these principles with existing development software business principles and approaches and he attempts to outline what is new or what is not.