Revisiting Software Engineering in the Social Era

Vanilson Burégio, Federal Rural University of Pernambuco, Racife, Brazil
Ejub Kajan, State University of Novi Pazar, Novi Pazar, Serbia
Mohamed Sellami, ISEP Paris, Paris, France
Noura Faci, Université Lyon 1, Lyon, France
Zakaria Maamar, Zayed University, Dubai, United Arab Emirates
Djamal Benslimane, Université Lyon 1, Lyon, France

ABSTRACT

This paper discusses the possible changes that software engineering will have to go through in response to the challenges and issues associated with social media. Indeed, people have never been so connected like nowadays by forming spontaneous relations with others (even strangers) and engaging in ad-hoc interactions. The Web is the backbone of this new social era – an open, global, ubiquitous, and pervasive platform for today’s society and world - suggesting that “everything” can socialize or be socialized. This paper also analyzes the evolution of software engineering as a discipline, points out the characteristics of social systems, and finally presents how these characteristics could affect software engineering’s models and practices. It is expected that social systems’ characteristics will make software engineering evolve one more time to tackle and address the social era’s challenges and issues, respectively.

KEYWORDS


INTRODUCTION

Software Engineering (SE) is a well-established discipline that examines the design, development, and deployment of software systems (or systems for short). SE addresses numerous questions like what phases to follow, what notations to adopt, what tools to use, what experiments to carry out, and what maintenance activities to plan. SE, also, helps ensure that systems satisfy users’ functional and non-functional requirements (e.g., reliability and availability). User involvement in system design and development is of paramount importance in any software engineering endeavor (de Waal & Batenburg, 2014).
Due to the continuous development of Information and Communication Technologies (ICT), software systems have gone through multiple changes. From monolithic blocks that are confined into specific places to distributed and heterogeneous blocks that spread out all over the world, and from systems that require human assistance when they breakdown to self-healing systems that fix breakdowns automatically. These are samples of changes that SE had to cope with. As a result, new SE paradigms have emerged and continue to emerge. The current century features new ICT trends like cloud/fog computing, big data, Internet of Things (IoT), and computer forensics that have triggered (sometimes “heated”) debates about SE appropriateness for these ICT trends (e.g., “addressing the software engineering challenges of big data”, software engineering institute’s blog, 2013). The Web, also, is another major trigger. Indeed, the Web now is the platform of choice upon which entire cross-border e-business systems are deployed. The evolution of the Web from 1.0 (readable online content) to 2.0 (editable online content) and then 3.0 (semantically rich online content) is a good indication of how ICT continue to drive the development of new systems. An example of such development is the emergence of social systems that are the focus of this paper.

Social systems display Web 2.0 (or social) era in which end-users simultaneously consume services and provide services (i.e., end-users are referred to as prosumers (Pedrinaci & Domingue, 2010) and are even anonymous but willing to help out). Investment on Web 2.0 (e.g., social networks, wikis, and blogs) is expected to reach $6.4billion in 2016, according to Forrester Research (Houston & Hoehler, 2013). Despite this heavy investment “... many large companies are embracing internal social networks, but for the most part, they’re not getting much from them,” according to Gartner¹. We attribute this limited return-on-investment to different factors including (1) unsuitability of some Web 2.0 applications for the corporate world (“is Facebook really a good tool for business?” open forum’s blog, 2009), (2) corporates’ negative perceptions (e.g., distraction) towards some Web 2.0 applications, and (3) over-emphasis on technology rather than humans. Could the inappropriateness of some SE current models and practices for developing social systems be another factor of this limited return-on-investment?

We define social systems as the result of wrapping independent Web 2.0 applications (e.g., Google+ and Twitter) so that they are exposed in a unified way to the world of organizations. The objective of this unification is to allow existing software systems (e.g., enterprise resource planning and payroll) to interact transparently with Web 2.0 applications regardless of their management policies and implementation technologies. Practically speaking, a social system will allow users (individuals and/or groups) of systems to integrate Web 2.0 applications’ services into their day-to-day operations. Examples of services could be tagging content and posting comment and examples of techniques could be crowdsourcing and mashup. Social systems will help organizations remain focused on their core competencies while “keeping an eye” on the social media. Our contributions include (1) a definition of social system along with its characteristics, (2) analysis of social system versus software system, and (3) rationale of revisiting some of SE well-grounded models and practices due to social systems’ characteristics. The rest of this paper is organized as follows. First we present the evolution that SE has gone through in terms of models, practices, and programming languages. Then we list signs of the social era such as building networks of contacts. We also list the characteristics of social systems and illustrate the response of SE to these characteristics. The use of social machines is an example of this response. Finally, we conclude with a set of recommendations that would frame revisiting SE in the social era.
Wiki for Agility
www.igi-global.com/chapter/wiki-for-agility/117301?camid=4v1a

Functional Testing Using OCL Predicates to Improve Software Quality
www.igi-global.com/article/functional-testing-using-ocl-predicates-to-improve-software-quality/126638?camid=4v1a