iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 3 **Issue:** V **Month of publication:** May 2015

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Software-Intensive Systems

Abhishek Jain[1], Manjeet Saini[2], Manohar Kumar[3]
Department of Computer Science, Dronacharya College Of Engineering, Gurgaon

Abstract: In this paper we present an overview of how can people produce and use Software-Intensive Systems. We will describe how different communities can across the Software Lifecycle. Finally we will focus on goals, challenges promising approaches, potential payoffs, Timeliness, costs and risks. The "Using Software-Intensive System" discussions concentrated on software engineering challenges in developing and improving what might be called "human intensive systems." These are systems in which people are participants in the execution of the system, not just as so-called users considered to be external to the system, but rather as integrated components of the system in much the same way as hardware and software components are related

Keywords: Software, cost, risks, timeliness, evaluation, end-user, human intensive system

I. INTRODUCTION

Computer-mediated communication technologies, such as email, audio/video conferencing, social networking, blogging, micro blogging, online discussion and question and answer forums, have transformed the way that communities connect online. Software development communities on many scales, from individuals, to teams, to organizations, and ecosystems of organizations, are able to take advantage of the ubiquity of these technologies to facilitate communication about, collaboration in, and coordination of shared work. For example, structured communication protocols can be encoded in online dispute resolution systems to enable more flexible, cheaper ways to work through negotiations than by meeting face-to-face. Free and Open Source Software (FOSS) development practices are conducted entirely online, using tools like mailing lists and open software repositories, to enable collaboration among diverse communities of constituents across distance, time, cultures, and even across projects in an ecosystem of related software (e.g., any one of the 20,000 projects in a typical GNU/Linux distribution). These free, open, online collections of software projects have indeed reached critical mass, enabling people from many different communities to innovate exponentially and create software products with little self-developed or maintained infrastructure support. Connecting software engineers together has the potential to hire more employees in software companies; to increase the sharing of timely, concise knowledge about current, attention-worthy development activities and events among communities of developers, testers, managers, those with non-engineering job roles, and consumers and customers of the software products; and to revitalize legacy software ecosystems whose component technologies have been made obsolete and whose participants have long ago moved on to newer endeavors.

II. GOALS

The goal is to enable various communities to easily, quickly, and economically create, maintain, and adapt software by understanding and improving how these communities of software engineers, supporters, and their users communicate, collaborate, and coordinate using various kinds of computer-mediated communication.

III. CHALLENGES

The communities range from collocated to globally distributed; from individuals to open source projects to industrial corporations; from developer-centric communities to multi-role communities including testing, specifications, management, marketing, and sales; from mainstream Internet-connected communities to those on the margins; and from currently active communities to those that are essentially defunct. Each community has its own set of stakeholders, problems, requirements for appropriate solutions, development tools, and varied means to deploy and enact solutions, requiring investigation, analysis, and realization approaches customized to each problem under consideration.

©IJRASET 2015: All Rights are Reserved

IV. PROMISING APPROACHES

The main research method consists of three steps. First, the researchers must understand the social context of the community under study. What development processes, work practices, communication modes, and collaboration and coordination norms do community members employ? Who are their stakeholders? What are their roles, whether formal or emergent? Who are the developers and the users? What is their culture? What are their values? What are their motivations? With whom do they interact to create, maintain, and adapt their?

Next, researchers must gain access to, gather, analyze, and validate software process information related to the software project. These steps should address the people as well as the software artifacts.

Finally, the project-related communications among the members of the community need to be compiled, including status meeting notes, bug triage records, code reviews, specification reviews, tags, annotations, blogs, public emails, online chat, microblogs, discussion forums, check-in messages, bug descriptions, and any other communication mode and forum that can be electronically recorded. Software development in the cloud makes it easier to collect and analyze these data sources.

Next, researchers need to perform multiple types of analysis to identify the critical connections between artifacts, activities, and communications. Communication logs can be used to understand how software engineers who are no longer working with a project conducted their development work.

These analyses are codified into tools designed to deliver relevant, concise, verifiable information to the appropriate audience, when needed. This can facilitate necessary and useful communication that can enhance collaboration and coordination among community members on various scales.

V. TIMELINESS

The communications and data availability environment have changed dramatically in the last few years facilitating people to produce and use software-intensive systems. Web 2.0 and other large-scale communication networks have become dominant only in the last five years; Facebook alone has 750 million users (more than one-tenth of the world population)! In addition, the scale of the communities and the data required for analysis is enormous, on the order of terabytes and petabytes for large individual projects and even small ecosystems of projects.

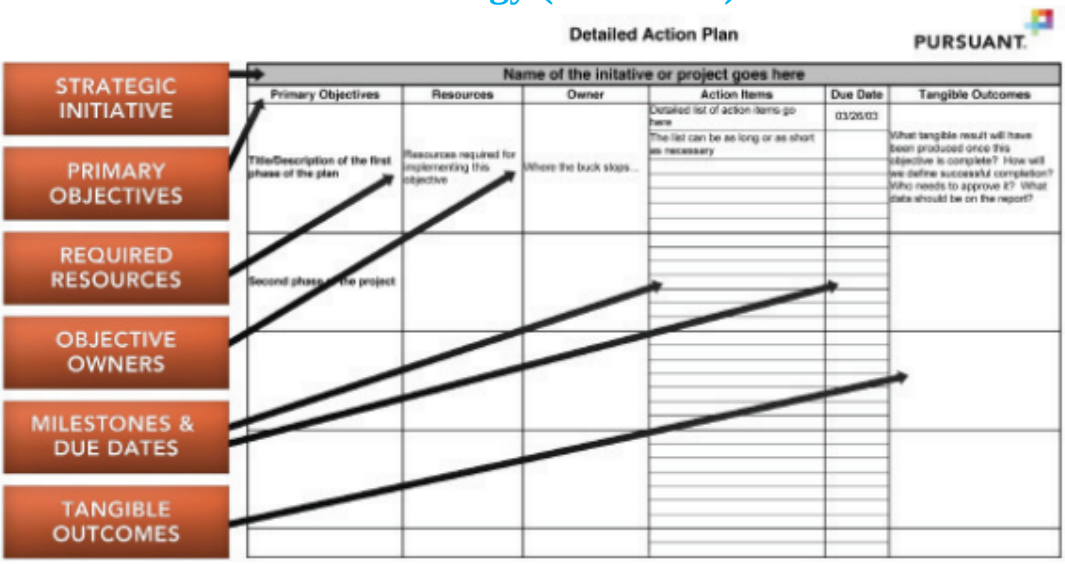
Today, a critical mass of publicly available, open source targeted components has been reached, enabling even lone developers to create entire software projects, requiring minimal additional design and programming.

VI. ACTION PLAN, JUMP-START ACTIVITIES

Many researchers are already working to understand and develop tools for various communities of developers. These activities include open source projects, software development corporations, scientists, medical organizations, defense contractors, school systems, and ecosystems of development projects and game platforms. The key next steps are to describe the relevant work practices of desired target communities, to identify their most relevant communication problems, and to design and build tools to address these problems.

©IJRASET 2015: All Rights are Reserved

Volume 3 Issue V, May 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)



VII. EVALUATION

Assessments are needed to determine whether the utility of the tools, the accessibility of the information, the engineering practicalities involved in the storage and analysis of data, and the means of using the results to facilitate communication and coordination among the people, are improving performance. In addition, evaluation metrics must be developed to demonstrate a return on investment (ROI) to prove that the tools (e.g., for the social network of the community, for the code, for their communications) directly improve the summative measures, such as team productivity, product quality, project cost, and project agility.

©IJRASET 2015: All Rights are Reserved

Volume 3 Issue V, May 2015
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

PERFORMANCE ASSESSMENT FORM (STAFF CADRES)					
PERFORMANCE ASSESSMENT OF Mr.				E.No.	
FOR THE PERIOD FROM		TO		DEPT.	
S.No	Factors for Assesment		Max Marks	Marks Awarded	JUSTIFICATION (Must)
1	JOB KNOWLEDGE		30		
2	INTEREST, INITIATIVE INVOLVEMENT & COMMITMENT TO JOB RESPONSIBILITY		30		
3	PERFORMANCE ON THE JOB	QUALITY	15		
		QUANTITY	15		
4	POTENTIAL TO PERFORM AT THE NEXT HIGHER LEVEL		30		
5	COMMUNICATION SKILL	ORAL	15		
		WRITTEN	15		
6	PERFORMANCE OF JOB WITH APPLICATION & PRESENCE OF MIND		15		
7	PLANNING, ORGANISING & CO-ORDINATING		20		
8	INCLINATION, ORGANISING ADDITIONAL RESPONSIBILITIES		20		
9	PUNCTUALITY & ATTENDANCE		30		
10	DISCIPLINE & MORAL		30		
11	RELATIONSHIP WITH	SUPERIORS	5		
		PEERS	5		
		SUBORDINATE	5		
12	ADHERANCE TO SAFETY, OTHER PRE-CAUTIONARY MEASURES		20		
TOTAL			300		
Recommendations : _____					

VIII. SOFTWARE ENGINEERING FOR END USER PROGRAMMERES

More than 10% of the world population is now using Facebook. These users engage in an elementary form of programming by specifying their privacy settings, namely “rule-based programming.” This is indeed a form of programming, because it is a way of instructing the computer what action to take when data arrives.

In the particular case of Facebook, any unintended effects of users’ “programs” affect not only their own privacy, but also the privacy of everyone connected to them.

Clearly, end-user programming (EUP) is an important phenomenon, and many people other than those who develop software as their profession also engage in this activity.

Examples include healthcare professionals, engineers, accountants, teachers, and even children.

Software development activities include creating new formulas in spreadsheets, customizing software with preference settings, “mash up” or combining Web services by dragging them together. Yet, the software engineering community has taken little notice of the enormous body of software these users are producing, and even less notice of how to help end-user programmers monitor, assess, and correct errors in the software they create or customize.

The overall goal of software engineering for end-user programmers is to introduce the software engineering research community to the end-user practices and environments to enable this new class of producer-consumers of software to more easily, quickly, and conveniently create, maintain, and/or adapt software with quality suitable for the software’s purpose.

©IJRASET 2015: All Rights are Reserved

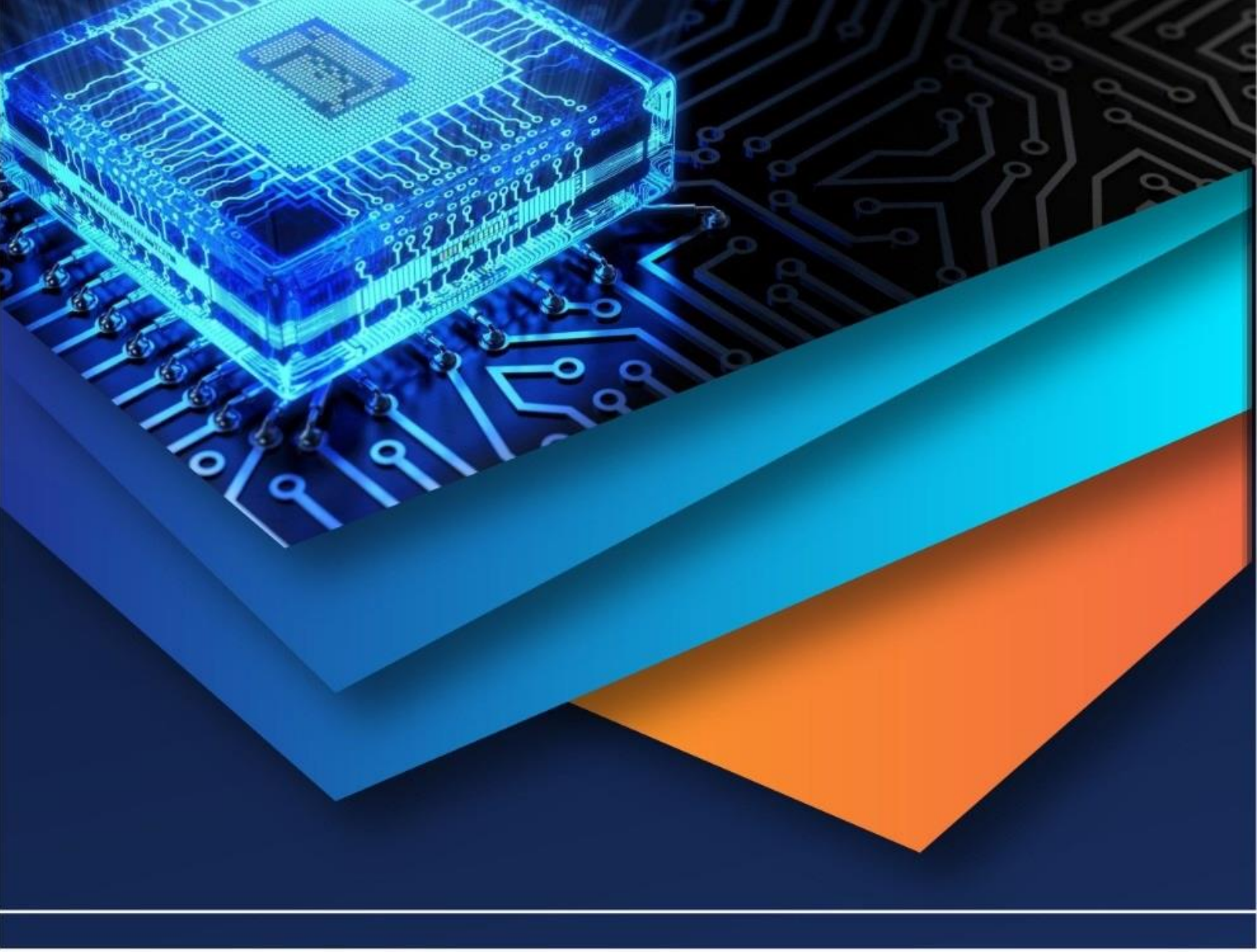
IX. CONCLUSION

The “Helping People Produce and Use Software-Intensive Systems” discussions concentrated on software engineering challenges in developing and improving what might be called "human intensive systems." These are systems in which people are participants in the execution of the system, not just as so-called users considered to be external to the system, but rather as integrated components of the system in much the same way as hardware and software components are related. Such systems are increasingly central to a large variety of domains.

REFERENCES

[1] Ambriola, V., R. Conradi and A. Fuggetta, Assessing process-centered software engineering environments, *ACM Trans. Softw. Eng. Methodol.* 6, 3, 283-328, 1997.
[2] Balzer, R., Transformational Implementation: An Example, *IEEE Trans. Software Engineering*, 7, 1, 3-14,1981.
[3] Chatters, B.W., M.M. Lehman, J.F. Ramil, and P. Werwick, Modeling a Software Evolution Process: A Long-Term Case Study, Software Process-Improvement and Practice, 5(2-3), 91-102, 2000.
[4] Cook, J.E., and A.Wolf, Discovering models of software processes from event-based data, *ACM Trans. Softw. Eng. Methodol*. 7, 3 (Jul. 1998), 215 - 249
[5] B. Curtis, H. Krasner, V. Shen, and N. Iscoe, On Building Software Process Models Under the Lamppost, *Proc. 9th. Intern. Conf. Software Engineering*, IEEE Computer Society, Monterey, CA, 96-103, 1987.

©IJRASET 2015: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)