Panel Session

Extreme RE: What if there is no time for Requirements Engineering?

Panel Chair: **Sol J. Greenspan**
Verizon Communications Inc.
Waltham, Massachusetts
USA

The term "Extreme RE," like "Extreme Sports," is supposed to conjure up images of people performing activities under novel and challenging conditions, where risks are inherent, and extreme measures are needed to overcome adversity, push boundaries, explore limits, or just get by. Imagine surfers seeking some of the world's highest waves, bike racing through deserts, golfers playing a course built on a frozen fjord in Northern Greenland. Well, maybe Extreme RE isn't quite as dramatic as Extreme Sports, and maybe we aren't seeking the same kind of adrenaline rush, but the analogy is there in spirit.

This panel will discuss how to do RE in the co-called "real-world" and, in particular, in the presence of severe time constraints. Increasingly, businesses are making time-to-market their dominant business driver, with other drivers such as cost and quality often taking a back seat. Anything that slows the delivery of the software is viewed as an obstacle to success.

For years we have been characterizing RE as an up-front activity whose goal is to determine requirements — correctly, consistently, and completely. We have warned that not doing enough RE is risky, since errors that persist after the so-called requirements phase will be orders of magnitude more difficult and expensive to detect and correct in retrospect. We generally advocate a controlled process of careful elicitation, comprehensive documentation, systematic modeling, formal analysis, among other time-consuming activities. We contend that it is worth the time and effort to do RE, and to do it well, and that it is worth the time to acquire, apply and improve RE methods, tools, and processes.

However, organizations under tight time (and budget) constraints are not finding our arguments sufficiently persuasive, and as a result, when time is tight, RE activities seem to be among the first ones to become neglected. Requirements-related activities are seen as time-consuming, labor-intensive, and hard to control and evaluate. If one listens closely, one can even find practitioners saying that each day spent on requirements makes the project another day late, whereas each day of coding is a day closer to meeting the deadline.

How can we adapt RE to what has become a much different, more competitive, and in many ways more extreme environment than the one that existed when we started rationalizing and justifying RE years ago.

Let's ask ourselves questions such as the following ones (which obviously could depend on the situation, as discussed later):

- Which activities of RE are essential to do even when the schedule is tight?
- How does time pressure alter how much, and what kinds of, information to record?
- How can a RE process (i.e., a set of coordinated RE activities) be organized and conducted to save time? Which tasks should be performed by the software developer and which ones by the business customer?
- What RE activities might be viewed as "overhead" on an individual project but could pay big returns over the course of a family of projects? How do you justify doing them on an individual project?
- How can we convincingly demonstrate and measure that the time actually expended on various RE tasks is well-spent?

To pursue the subject in more depth, we need to review the reasons for doing RE in the first place and judge how essential, or dispensable, we find each reason. We can then respond accordingly by omitting the associated tasks, scaling down with a light-weight approach, changing the timing, or applying tools, depending on the nature of the task. Here are some typical reasons for doing requirements:

Proceedings of the Fifth International Symposium on Requirements Engineering (RE'01)
1090-705X/01 $10.00 © 2001 **IEEE**

- As the technical basis for a contract or statement of work
- To explain to developers what they need to build, and check that they have built it
- For business stakeholders to understand their problem and feasible solutions, including interactions between stakeholders and between systems
- For project management to plan/review/control progress
- To specify the details of features, functions, user interfaces, components, DB design, etc.
- To generate test cases
- To comply with standards, e.g., for process certification, or legal requirements

Let's take the first reason as an example of a point of discussion. In order for software development to be a manageable enterprise, there needs to be a technical basis for a contract, so that software could be managed like any other type of product. This is especially true for government spending, where the bidding and accountability procedures need to be explicit and publicly defensible. However, in order to streamline their business relationships, some business partners may decide to simplify the contractual relationship. If they substitute a little trust for some of the legal contractual relationship, can they save some time? If they still want to do a rigorous job of RE for other purposes, do they still save something by taking a lighter approach to the contract? And under what circumstances is this approach applicable?

Also taking advantage of the "extreme" label is a software development approach called Extreme Programming (XP), whose principles are aimed at coping with, among other things, the time constraints that pressurize and threaten failure of so many software projects today. (I confess that it was XP, not sports, that stimulated the title of this panel.) XP takes a fresh, and some would say extreme, approach to software development. Among other things, XP prescribes how to handle requirements in the circumstances where XP is applicable. As such, XP serves as one interesting example of how to deal with requirements under time pressure. In the panel discussion, we will learn what XP recommends to deal with requirements, and we will make sure that the conditions of applicability are specified (e.g., small teams working closely with the customer to develop software quickly despite rapidly-changing requirements).

We will also characterize other stereotypical situations where time is critical and propose how to meet the needs presented by those circumstances. For example, in a large company, where XP , the key concerns might include such things as process modeling, knowledge management, and vendor management, and the recommendation of XP might not apply at all.

Yet other situations are those that entail quality attributes such as safety and security have priorities that vie with the time constraints. The software might be needed yesterday, but it might also be unacceptable (or against the law) to field a product or service that fails to meet certain safety or security requirements.

Any recommendations for how to do requirements under time constraints should parameterize the circumstances of applicability in terms of such things as system size, team size, legacy vs. new development, product family vs. one-of-a-kind, waterfall vs. prototyping, centralized vs. distributed application, centralized vs. distributed development, other characteristics of application domain.

The panel discussion could also address under what circumstances investment in other resources can help achieve time-saving objectives. For example, keeping a repository of requirements information might be a costly commitment in terms of staff, tools and organizational overhead; however, subsequent projects might be significantly accelerated, making the investment ultimately worthwhile. Similarly, a business might find that by maintaining up-to-date business process maps depicting the current mode of operation, it can more quickly write the requirements it provides to subcontractors. Can an investment in a well-defined and maintained systems architecture contribute positively to meeting time deadlines?

Other questions to consider are the following:

When requirements change a lot, it can be difficult to stay on schedule. When does it work to "freeze" requirements, and when is freezing the requirements impractical.

What kinds of requirements tools will help in meeting time deadlines?

What kind of personnel, expertise, training, knowledge base, organizational structure is best for providing time-effective RE services?

Acknowledgment. *Thank you, Annie Antón, for helping to get this panel off the ground.*