Software Engineering Must Be Collaboration-Aware

Norbert Jastroch  
MET Communications GmbH  
Eschbacher Weg 10  
61352 Bad Homburg, Germany  
norbert.jastroch@metcommunications.de

Vassilka Kirova  
Alcatel-Lucent  
600-700 Mountain Ave.  
Murray Hill NJ 07974, USA  
vassilka.kirova@alcatel-lucent.com

Cyril S. Ku  
Department of Computer Science  
William Paterson University  
Wayne, NJ 07470, USA  
kuc@wpunj.edu

Thomas J. Marlowe  
Department of Mathematics and  
Computer Science  
Seton Hall University  
South Orange NJ 07079, USA  
thomas.marlowe@shu.edu

Mojgan Mohtashami  
Advanced Infrastructure Design  
1 Crossroads Drive  
Hamilton NJ 08691, USA  
mojgan@aidpe.com

ABSTRACT
As software development becomes more collaborative, all aspects of software engineering and their management need to accommodate and support collaboration. In this paper we briefly survey key concerns, known challenges, and potential alternative solutions.

Keywords: Software architecture, requirements, collaboration, project management, agility, components, interfaces, knowledge management, business policies

1. INTRODUCTION
Inter-organizational cooperation in software product and tool development is of increasing importance and great potential. Continuing trends for outsourcing and offshoring, subcontracting, and academic-industrial collaboration, plus increasing sophistication and specialization of business software (and for that matter, most of knowledge-intensive software), make an ever stronger case for close inter-institutional collaboration throughout the development process. While legal issues and management support appear to be the most substantial challenges, we contend that all aspects of software development need to be re-examined and revised to address collaboration. In this paper, we consider collaborative development of large, complex systems, with large or complex components, which may however be feature- or service-based, rather than resulting from a functional decomposition.

A key challenge in multi-organizational collaboration efforts is that institutional and team responsibilities need to be assigned early in the project inception phase which requires the scoping, decomposition of the product and the definition of the overall project structure to be established well before the application and often the technology to be developed is even marginally understood. That, especially in large, complex or innovative projects, in turn requires flexibility in assignment of responsibilities and in interactions between teams and partnering institutions. It further requires flexibility in defining the scope and allocation of responsibilities to the components, subsystems or features they are to deliver. Given the collaboration constraints it is crucial to understand that the best high-level decomposition may not be the optimal structural or functional decomposition, but will also be driven by management objectives, technical expertise, resources, and other considerations in the individual organizations. Any tradeoffs evidently should not devalue the collaboration below any established cost/benefits threshold or jeopardize the system objectives. This is a critical consideration in assembling the collaborative venture at the start.

A notable current trend in both the technical and management aspects of software development is agility. Agile frameworks accommodate changing requirements through continuing customer involvement, self-organizing closely collaborating cross-functional teams, short iterations, application of technical practices such as test-driven and acceptance test-driven development, test automation and continues integration, as well as through avoiding unnecessary specification and upfront extensive analysis. Agile methods bring the much needed flexibility but provided the constraints of multi-organizational collaboration we expect that a typical collaborative development effort will support local agility and controlled flexibility at organizational and corresponding component and subsystem boundaries and interface.
In the body of the paper, we briefly consider a number of policy, artifact, and process issues that need to be resolved to foster and optimize both the eventual products of the collaborative venture and the health of the collaboration itself. Addressing these issues complements rather than competes with selection and establishment of a good collaborative tool environment. Tools address the question of how collaborative work is to be supported, whereas our concern here is focused on what needs to be done to allow the process, product and partnership to succeed.

2. EARLY BUSINESS POLICY DECISIONS
In a collaborative software development project, several key decisions must be made prior to anything but the earliest vision of a system/product and its business case. The first, of course, is determining whether to collaborate, how to collaborate, and with whom [16]. Assuming a fully collaborative project, decisions must be made in several dimensions, before or during project inception: responsibilities in the collaboration, resource acquisition and management, establishing or reinforcing trust and familiarity, process and platform consistency, and protection of partner interests [17].

Responsibility and resource issues include (1) a policy on sharing of resources, including key personnel, and on cross-organizational discipline-specific collaboration [3], and (2) responsibility for the cost of new shared tools, personnel, and resources. Business project support includes (1) establishing risk management plans and management contingency policies for the collaboration, and (2) establishing a venue or process for mediation and arbitration, as well as (3) creating inter-organizational plans for cross-sensitizing, communication and training, and (4) establishing a procedure for interaction with the customer and other stakeholders—who should not be confused and overwhelmed by multiple contacts with different organizations.

Process and platform technical support includes (1) standardization of platforms, tools, and processes—or at a minimum alignment and support of standard and consistent views, (2) creating a shared software configuration management environment, and guidelines for sharing of artifacts, and (3) determining a level of agility and a level of formality—locally and at interfaces [5, 8, 14]. Finally, collaboration must be supported by seeking and maintaining steady buy-in from management, technical staff, IT and legal departments, while respecting the interests of the individual partners and stakeholders.

3. REQUIREMENTS AND KNOWLEDGE MANAGEMENT
Incomplete, imprecise or changing requirements interact with the initial architectural decomposition, affected by missing knowledge, dynamic information, or the need for knowledge integration and synthesis. Hence, means for iterative discovery, analysis, knowledge building, and delivery become essential. First, support for dynamism in defining features and components is needed [20]. Second, approaches must be developed for collaborative knowledge discovery and integration, and collaboration in detecting and proactively managing unknown risk (compare [15]), while addressing risks to security, privacy and intellectual property from this coordination. Third, complex systems working within dynamically changing environments are influenced by inbound influences and outbound impact—beyond their functional interfaces—that cannot be fully anticipated, but to which they need to adapt. These inbound and outbound factors may introduce risks or opportunities, and may be well, or partially, or even not at all understood in advance. As result, support for adaptability becomes a major operational requirement and consideration.

4. ARCHITECTURE, ANALYSIS AND DESIGN
The large-scale organization of the product—its architecture—is partially determined by (or along with) the high-level decomposition and definition of interfaces between components developed by different parties—compare [7]. Although the parties could use different architectures within their own components, uniformity (or at least consistency) of architecture, of process and of development platform increases communication bandwidth through shared vocabulary and mental maps and reduces miscommunication; supports better software configuration management and flexible interfaces, facilitates knowledge discovery and integration; and enables evolution, scalability and other non-functional properties. Most collaborative ventures will benefit from a scalable architecture with some measure of agility within components and flexibility across their boundaries.

Agile development practices, if used, will have impact as well. [2, 5] suggest approaches for extending agility to widely distributed and collaborative ventures. Assuming an incremental iterative (if not fully agile) collaboration requires changes both in artifacts and in the activities. The most important from an architecture and design point of view is that there will be two distinct if sometimes overlapping workflows: one, determining high-level components, boundaries and interfaces, and a second one determining the internal organization of those components, both of which can run in iterations gradually addressing additional aspects, features and requirements or adding more details.

Inter-component interfaces must be flexible enough to support at a minimum handling of newly discovered exceptions and alternate flows, and to permit exchange of metadata and system information [8, 13, 14] for cross-component optimization and refactoring [4], without initially requiring communication of large amounts of such information, only a small fraction of which may actually be useful. Such interfaces can be extended to process and business objects, to support other activities and artifacts such as cross-component traceability and policy localization/specialization. On the other hand, the partition into components cannot be considered “sufficient to proceed” without at
least some analysis of component responsibilities. These cannot be determined without considering both structural factors such as internal cohesion and cross-component coupling, in order to limit the cross-organizational footprint of change, and policy and resource factors such as division of labor and availability of expertise.

5. TESTING

Testing is inherently a collaborative activity [21]. It depends on close collaboration between teams and team members and often involves interactions with customers. Thus policy and governance rules similar to the ones related to requirements apply and need to be addressed accordingly.

Testing is ongoing, multilayered (unit, integration, acceptance) and should be automated. The organization of testing activities highly depends on the project partitioning model, the process models (development frameworks) used by the individual partners, the architecture of the product, the organization of the collaborative partnership, and the allocation of responsibilities. Unit level testing is essential and remains a local responsibility, while the responsibilities for integration and acceptance testing have to be agreed on as part of the collaborative partnership. Integration testing across organizational boundaries may require additional planning, testing effort, coordination, and sharing of testing strategies and resources. Not only does integration testing have to work across organizational boundaries, but in addition responsibilities for debugging and fixes must be assigned across those boundaries. Further, if more sophisticated and stateful interfaces or interaction patterns, such as those in [8, 13] are used, then interfaces themselves must be subject to unit testing. Also regression testing for changes that affect interfaces and possibly cross component boundaries will only be as effective as cross-component dependence analysis and traceability allow [8]. Acceptance testing reintroduces the issue of customer and user interaction and of indirect contact with the customer.

Any selected testing model will place specific requirements on the development, build (including continues integration), revision control and change management environments. For example, continuous or early system and stress testing will require continuous integration or early integrated builds of the entire product. The effectiveness of the testing will depend not only on test automation, but also on good global revision control, change management, and traceability tools.

The testing practices have to be well-aligned with the requirements and feature definition and scoping methods and will depend on the development approach. And while full deployment of agile frameworks such as SCRUM [9, 10] across all partners of a multi-organizational collaboration project may not be feasible (due to organizational management, legal and other constraints), implementing agile test-related engineering practices is arguably a good fit. For instance, ATDD (Acceptance Test Driven Development) [10, 18] provides for good ongoing collaboration within the teams as well as with the customers.

6. METRICS AND EVALUATION

There are three major issues for collaboration-aware metrics: the selection of metrics, the gathering of data (measurement) for the metrics, and interpreting the resulting measures. Just as testing consists of unit testing, integration testing, and system/project testing, metrics for a collaborative project will include component metrics, organizational metrics (since an organization may be developing more than one component), and global metrics, for product, project, and process alike.

Evaluation for and within collaborative systems requires metrics that (1) accommodate collaboration, (2) measure readiness for collaboration and the effectiveness of the collaboration, and (3) identify business policy/process and software process obstacles, and driving forces and enablers for collaboration. For accommodation, existing metrics (whether used locally or for the entire project) must be modified or re-interpreted, and new metrics may be needed. For example, effort and resource estimates must account for the impact of collaboration. On the other hand, cost estimates and measures can be used to evaluate the effectiveness and overhead of collaboration, and also to guide project restructuring and system refactoring. Still other existing metrics may no longer be useful in a collaborative setting, or may need to be retired or replaced.

Readiness metrics must measure not only technical readiness, but management support for collaboration. Finally, while data collection is not a serious problem for local metrics, comparison of measures or computation of global metrics necessitates development of guidelines and practices for uniform collection. Responsibility for collection of the relevant information must be assigned (to an organization or a tool), and consideration must be given as to the proper encoding and weighting of the resulting measures.

7. MAINTENANCE AND EVOLUTION

Post-release responsibilities for maintaining a large, complex and strongly interacting system raise new issues for collaborative development. First, the lifetime of the system or product line may be longer than the lifetime of the collaboration. Also, a single change may require small changes in a large number of components. Finally, there is a timing issue: other than for major faults or security threats, changes can often be deferred. Key questions are (1) Who is responsible for determining that a change is needed, and when the change should be made? (2) How will the nature and locations of the needed changes be determined? and (3) Who will be involved in
implementing and in testing the changed system? These questions tie into both the need for global traceability/dependence analysis [8] and arbitration [15].

Perfecitve maintenance, as well as reuse, often involves refactoring. Local refactorings, assuming flexible interface structures (e.g., Façade and Adapter design patterns), is typically a local matter. But refactoring across interfaces or changing component responsibilities will again involve negotiation and determination of responsibilities.

8. IT AND COMMUNICATION SUPPORT
Formal and informal communication links between partners form one of the four key factors in collaborative success, after management and technical competence, collaboration-aware risk management, and a collaboration-friendly management and technical environment. The success of any collaborative venture depends on availability of multiple forms of communication and meeting support, and technical and managerial support by IT departments and staff. The development environment should provide for sharing of code, test structure, design artifacts, and so on, preferably in shared tools or views. The Cloud and virtual environments provide additional opportunities and resources, with some associated risks.

IT is also responsible for implementing security, access control, intellectual property and privacy protections, knowledge management, etc. Typically a layered or hierarchical permission structure will be required, allowing sharing of some information (or summaries) with collaborators but not outside world, as defined by rules and policies in the collaboration agreement.

9. PROCESS COMPLIANCE AND OPTIMIZATION
Both product and process in software development are frequently held to standards, whether internal, industry-based, or required by the customer or by regulation. In the first three cases, both product and process compliance can follow a structure much like that we propose for risk management [15]: Internal compliance checking by each party, mutual checking at interfaces, and a mechanism for arbitration and negotiation. In some situations, review by a certified third party may also be required. Additional issues include: obstacles arising from internal standards, process or artifact inconsistency, and compliance scope across national boundaries. Finally, technical process and business policy optimization resembles perfective maintenance—very important and usually beneficial, but neither urgent nor worry-free, and therefore requires a similar process.

10. BUSINESS PROCESS ISSUES AND MANAGEMENT CONTINGENCY PROCESSES
Business management in a collaboration handles the standard tasks for management of software projects—personnel, budget and schedule tracking, and so on. Beyond these, in the literature and in our previous work [15, 17], the most significant policy issues for collaboration are identified as (1) creating and maintaining trust, (2) handling differences in language and culture, both organizational and social, and (3) maintaining corporate support. In addition, management processes must assure the continuing quality of risk management and communication, and deal with problems and changes in the set of partners and with non-fulfillment of partner responsibilities.

11. SECURITY, PRIVACY, AND INTELLECTUAL PROPERTY
Collaborative processes need knowledge from diverse sources, some of which raise specific security, privacy, or intellectual property concerns: product information, process information, platform and tool information, and recent corporate decisions and history. Use of such information, and more general concerns about such issues tend to harden management, IT, and legal expert resistance, and to inhibit collaboration. These issues must be addressed along both corporate (social/economic/legal) and technical dimensions.

On the corporate side, consider cost-benefit analysis for various levels of information sharing, with restrictions on external use of information gained. Sharing promotes trust and cooperation, with the risk of high rewards for low contributions. This suggests a differential approach, in which information is layered, and inner layers revealed only as a partner contributes. However, some knowledge must be shared a priori, since it will be required to initiate the collaborative process or product inception. In addition, changes in the set of partners can pose difficulties for this approach.

Technically, there are at least two aspects in alleviating these problems: first, selection or development of filters, abstractions, or views of information so that useful but safe summaries are available to collaborators and customers; and second, selection of publicly available or sharable tools and methods through which information can be imported and exported. These of course must complement use, undertaken and certified by all collaborators and other stakeholders, of standard secure mechanisms for data storage and information transmission, to address third-party threats. This is especially important for the development and communication platforms. Successful collaboration must rely on a shared and uniform view of important artifacts, and on rapid, reliable, and secure broad-spectrum communication, both formal and informal.
12. RELATED WORK AND CONCLUSIONS

Most of the existing literature on collaboration considers intra-organizational collaboration, or focuses on selected aspects of inter-organizational interaction. This paper in contrast specifically addresses inter-organizational collaboration from a broad, multi-faceted and systemic point of view. Incorporating agile practices in intra- or inter-organizational collaboration is discussed in [2, 5, 6, 11, 12]. Herbsleb [7] and Whitehead [21] address the problems of collaboration, but largely within a single organization, and primarily limited to tool support and software configuration management, including expansion of the set of desirable artifacts; changes to management policy and perceptions, and in the software development process, are also discussed. Erickson [1] and Schadewitz [19] provide patterns for component interfaces and interactions. Excluding our previous work, there is limited explicit focus on collaboration-aware metrics, changes in testing, knowledge management for collaborative software development, or risk management, areas we briefly address in this paper.

In conclusion, we have addressed the implications of inter-organizational collaboration for a number of development aspects—including both core software engineering and umbrella activities. Inter-organizational development and collaboration for large, complex or innovative software products calls for new approaches to accommodate both organizational differences and flexibility in development. As we discuss, all aspects of software development—technical, process, and management—are affected. This paper outlines pressing issues and presents initial or partial solutions in several areas.

References