Software Process warm-up

Q1: Die Beschreibungen von Software Engineering Prozessen nehmen ein "green field" Szenario an, d.h. dass ausschließlich neue Software zu entwickeln ist. Viele Projekte im IT Bereich drehen sich um das Ersetzen eines in-house entwickelten IS durch eine ERP (Enterprise Resource Planning) Lösung. Die Schlüsselwörter hier sind „customization" und „integration". Sind Software Engineering Kentnisse hilfreich in dieser Situation?

Q2: Business Process Reengineering (BPR) ist eine Methodologie durch die die Arbeitspraxis und die Betriebe in einer Organisation von neuem entworfen werden, ohne die vorhandenen Strukturen und die Verfahren zu beachten. Nehmen wir an, dass ein BPR Projekt an einer Kommunalregierung durchgeführt wurde. Die daraus resultierenden Empfehlungen beruhen stark auf IT. Software Ingenieure werden gefragt, wie die Empfehlungen zu implementieren sind. Was würden Sie in dieser Situation sagen?

Q3: Eine Neuentwicklung auf dem Softwaretechnikgebiet ist MDD (Model Driven Development). Dieses behauptet, dass wir Werkzeuge benötigen, um ein Modell eines Software-Systems zu konstruieren, sodass die Modelle automatisch in funktionsfähig Software umgewandelt werden können. Produktivitätsgewinne würden auftreten, weil Modelle abstrakter und verständlicher sind als Quellenprogramme, Deploymentdeskriptoren, usw. Zudem erhebt MDD den Anspruch, der folgende logische Schritt in der Geschichte von Programmiersprachen zu sein: 0s and 1s, Assembler, C, Java. Nehmen wir an, solche Werkzeuge sind einsatzbereit. Was würde die Auswirkung auf Software-Prozesse (z.B. auf dem V-Modell) sein?

Process models targeting component-oriented development

In "A Process Model for Component-Oriented Software Engineering" (IEEE Software, Vol. 20, No. 2, pp. 34-41) behaupten Dogru and Tanik, dass ein spezifisches Prozessmodell wegen der qualitativen Unterschiede zwischen Komponenten und Objekten erforderlich ist:

> *"Our research is aimed at purely component-oriented software development as opposed to component-based approaches. Component-based approaches are usually object oriented but can also represent components ... The main question is whether to develop (a component-based approach) or to integrate (a component-oriented approach)"*
>
> *A remarkable difference between the object-oriented and component-oriented approaches is the use of inheritance… A combo box component, for example, could inherit from a list box. Once the combo box is a reusable binary component, however, the system is only interested in its interface definition, ignoring how it was built."*

Die Verfechter eines ähnlichen Prozesses für komponentenorientierte Entwicklung (www.umlcomponents.com) nennen die Eigenschaften, die ein Komponentenmodell bilden:

- *defined set of services that support the software (e.g. remote access, transactions, persistent storage, security; typically used by configuration and not by explicitly invoking those services as was done in CORBA)*
- *set of rules that our software must obey to interact with other components or with the runtime environment that hosts the components*

Beispiele für die Richtlinien, die im letzten Punkt erwähnt werden, sind in der EJB Spezifikation zu finden: *"EJB components should not access static fields", "EJB components should not use synchronization primitives"*, usw. Sie garantieren, dass keine unerwünschte Konflikte oder Wechselwirkungen mit dem EJB container auftreten, in dem die Komponenten laufen.

Dieser Prozess schließt für Spezifikation der Komponenten die Schritte ein, die in Figure 1 gezeigt werden:

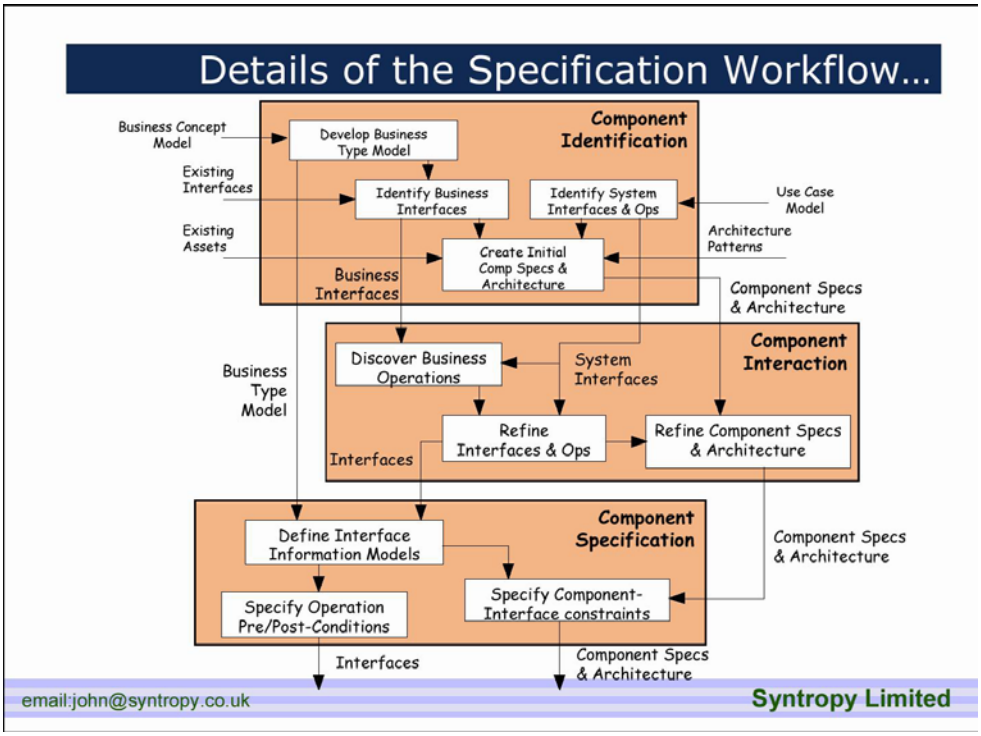

Figure 1: Specification phase in the UMLComponents process

Nach diesen Betrachtungen:

Q4: Sind Sie der Meinung, dass ein neuer Software-Prozeß erforderlich ist, wenn das Software Projekt hauptsächlich von „component integration" handelt? Warum?

Q5: Was würde passieren, wenn ein Software-Entwicklungsprojekt Gebrauch von Web Services macht? ("Service Oriented Architecture") . Welchem Softwareprozess würden Sie in diesem Fall folgen? Warum?

Q6: Übrigens, wie sollte ein Software-Prozeß definiert werden? Genügt ein Artikel oder wird eine längere Erklärung benötigt? Sollte die Prozessdefinition eine Darstellung (d.h. graphische Modellierungssprache) auch vorschlagen?

Q7: Die meisten Prozesse legen einen grossen Wert auf Disziplin. Können Sie an andere Artefakten denken, die unvermeidbar/notwendig/nützlich seien, um der Gewinnung des maximalen Nutzens aus einem Prozess zu versichern? Werkzeuge?

Q8: Nennen Sie Informationsquellen (web sites, conferences, organizations) die sich mit Software Engineering beschäftigen.

Process Improvement

Das Software Engineering Institute an der Carnegie Mellon University (www.sei.cmu.edu) treibt Forschung und Technologietransfer auf dem Gebiet der Softwaretechnik. Neue Beiträge dieses Instituts konzentrieren auf *process improvement*: die Maßnahmen, die Organisationen anwenden können, um den Reifesgrad ihres Software-Entwicklung Prozesses festzustellen, sowie die Techniken, die diese Organisationen anwenden können, um solche Prozesse durchgehend zu verbessern.

Drei Bereiche für Verbesserung sind vom SEI gekennzeichnet:

- Personal Software Process (PSP): "Process Improvement at the individual level"
- Team Software Process (TSP): "Process Improvement at the team level"
- Capability Maturity Model (CMM): "Process Improvement for organizations"

Ein Kommentar betreffend Teamarbeit:

> *The success or failure of a project is seldom due to technical issues. You almost never find yourself asking "has the state of the art advanced far enough so that this program can be written?" Of course it has. If the project goes down the tubes, it will be non-technical, human interaction problems that do it in. The team will fail to bind, or the developers will fail to gain rapport with the users, or people will fight interminably over meaningless methodological issues.*
>
> *Tom DeMarco*

Humphrey, der bekannteste Autor über TSP, liefert eine Liste der Probleme, die allgemein in dem Team Level beobachtet werden können: erfolglose Führung, Failure to compromise/cooperate, Mangel an Teilnahme, geringe Qualität, Funktion Ausdehnung, Ineffective peer evaluation

Q9: Wie kann man diese Probleme diagnostizieren, und wie kann man sie lösen?

In TSP ist es notwendig, Informationen während der Durchführung einem Projekt zu erfassen (*"you cannot improve what you cannot measure"*). Als Beispiel wird ein Excel Worksheet in Figure 2 gezeigt, um die Zeit zu notieren, die die Teilnehmer in ihren Tätigkeiten brauchen.

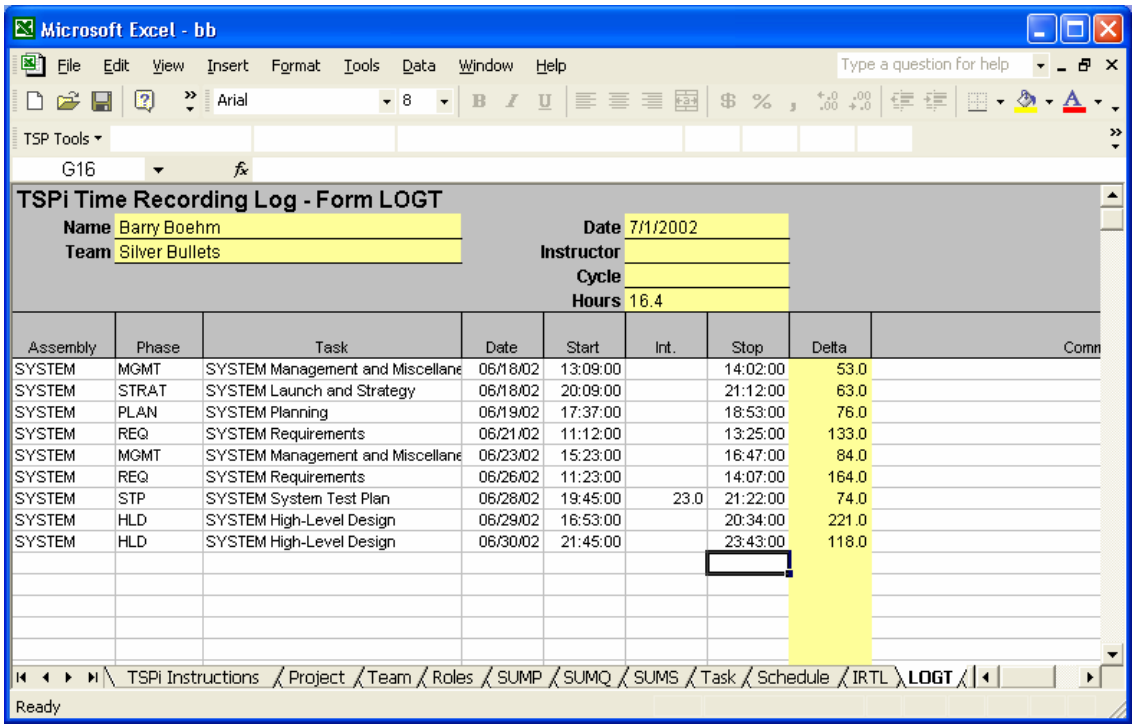

Figure 2: Time Log in TSP

Q10: In einem Software-Entwicklung Projekt müssen einige Produkte hergestellt werden (Quellcode, Doku, Training Material, etc). Die Herstellung von zusätzlichen Dokumenten, wie die Prozessinformationen die TSP erfordert, muss Vorteile und Nachteile haben. Unter welchen Umständen denken Sie solche zusätzliche Arbeit könnte unproduktiv sein?

Software Quality

Ein Dialog zwischen Q (Questor, einem Student) und seinem Mentor (A, dem Autor von [1]) geht um Software Quality:

> *Q: Sometimes the distinction between what's in the machine — and under our control — and what's in the environment — and out of our control — is tricky.*

A: Indeed. For example, imagine you're creating software to control an elevator. The customer would probably insist that the elevator car only stop at points where there is a floor — not between floors two and three, for instance. Would that requirement be part of the specification?

Q: If we're just writing the software, then the elevator car is actually just part of the environment. Our software simply sends signals that — hopefully — translate into correct actions by the elevator. So, I guess that requirement wouldn't be part of the specification.

A: Good. We software developers don't have any control over the elevator car itself; that's the domain of mechanical engineers. So we can't really guarantee anything about its behavior.

Q: Our software may conform perfectly to the specification, and the elevator could still do bad things: the mechanical controls might not be hooked up to our software controller, the power might be disconnected, the elevator car might be physically jammed, and so on.

Q8: Denken Sie an ein Beispiel auf dem Business Software Gebiet, in dem ein Qualitätsmerkmal nicht durch Softwaretechnik allein versichert werden kann.

Additional Resources

Ausführlichere Analyse von dem Team Software Process sind unter http://www.sei.cmu.edu/tsp/ zu finden (Powerpoints, Papers, und Video)

Beispiele der Probleme und der Lösungstechniken rund um Validation und Verifikation können auf dem Gebiet der Kommunikationsprotokolle gefunden werden:

- Design and Validation of Computer Protocols, http://spinroot.com/spin/Doc/Book91.html
- The Model Checker Spin, IEEE Trans. on Software Engineering, Vol. 23, No. 5, May 1997, pp. 279-295. http://spinroot.com/spin/Doc/ieee97.pdf

Ein Kurs über Modell-und Programm-Überprüfung wird vom Prof. Hatcliff at Kansas State University (http://www.cis.ksu.edu/~hatcliff/771/) angeboten.

Model Driven Development, by Greenfield, Short, Cook, and Kent: www.softmetaware.com/oopsla2003/greenfield.pdf . Sowie das Buch:

- Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. ISBN: 0-471-20284-3

[i] C. Wallace and J. K. Huggins. ASM 101: An Abstract State Machine primer. Technical Report, http://www.cs.mtu.edu/~wallace/pubs/primer.pdf