

# Taking Off to the Smart Shore – Part I

By Jochen Krebs  
Oct 23, 2006

**Summary:** Offshore and onshore development, outsourcing, and the relocation of IT jobs to low-cost countries are headlines we watch in everyday news. During Jochen Krebs' career, he has had the chance to play several different roles in software engineering, and through that experience he shares some thoughts and ideas about current industry trends. The scenarios in the first section of this article, "Inside the Project," will lead to the second, which offers a critical view on those trends titled "Going Abroad." The last section, the "Smart-Shore," addresses trends for a better offshore approach.

## Part I – Inside the Project

Part I of this two-part series takes a close look at the existing challenges in software projects that have not yet been delegated offshore. The primary focus is on topics that will be used to discuss the issues in developing offshore software systems. Conclusions at the end of each section will help summarize the challenges that are often addressed by agile software engineering techniques.

### Coding vs. Solution

When I was a programmer, I once faced a situation where I could not come up with an elegant solution to a particular programming challenge. Even though the scope of the issue was small, thinking about a solution for three days straight without any success wore me out. Then, it hit me like lightning--I had the answer! Immediately, I captured the thought, rethought the entire solution overnight, and wrote the code the following morning within a few minutes. I solved the problem, and it was out of my way forever. But how long did it take me: just a few minutes, or three days and a few minutes?

Writing good systems is not about "typing skills" and often it's not about the syntax of the programming language. Even though programmers need to know the capabilities of their programming languages (e.g., Cobol or Java), using one language over another doesn't guarantee success. For example, do we care about the programming language behind Microsoft Word, or how many lines of code it took to write the program? Not really, because customers often make buying decisions based on features.

Tools are a great selling point. For example, if you decide to write a novel, you will need a tool in the same way the programmer needs a programming language. Let's assume Microsoft Word provides the features the writer is looking for. What do you think are more important skills for the success of his final product: typing skills or writing and language skills? Would anybody care if a good author, such as Hemingway, is a fast or slow typist?

*Conclusion: Programming is more than typing.*

### Interpersonal vs. Intercultural

Building software is about the three Ps: people, people, and people. Nothing can compensate for having the wrong people on the team.

If you build software for customers, translating and materializing their ideas and even exceeding their expectations is priority number one. Interpersonal and facilitation skills dominate this discipline in the software engineering process. During analysis, design, and programming, your team members need to work together like a well-oiled machine. They must discuss various design decisions, communicate challenges and solutions, and harmonize in terms of working hours. Software development is about collaboration, which requires interaction among technical and non-technical folks.

Soft skills combined with technical expertise are also ingredients for success. Software engineers need to present solutions, convince, negotiate, and promote ideas. This is already a tough job within the same cultural boundary. When software development goes abroad, cultural differences, gestures, words, and expressions can easily change the meaning and dynamics of your project. Global software development requires sensitivity toward culture and tradition.

*Conclusion: "P" stands for people, not person.*

### The Value of Software

What mental picture do you conjure when someone says "Mercedes-Benz?" I personally think about

reliability, quality, safety, and luxurious driving behavior. I actually think about the biggest car in the fleet with the best-packaged equipment, although they build much smaller cars. Perhaps your personal experience is different, but on a large global scale, the brand and the reputation is considered to be high quality.

The car manufacturer not only is able to consistently build high-quality cars but also can put a hefty price tag on them, which attracts wealthier customers. Why not follow the car manufacturer's model and build high-quality software? Is it possible to create brand loyalty in software development?

Software can be used and copied as often as we like, if we do not consider the legal implications, of course. Software does not need regular maintenance, but does occasionally require updates that are packaged in service packs and patches. Tightening screws every 50,000 miles and dealing with rust in cars is expected, yet those additional software releases are often seen as fixes to "errors" in the original product.

Maintenance of old software became obvious during the Y2K change, when organizations needed to make changes to software that had operated for the past thirty years without any modification. Some of those programs were so reliable that people forgot they even existed. Some items actually increase in value over time, like old cars and other vintage items. But who would be proud to run MS DOS on Intel 80286?

A good example of things that increase in value over time is the Internet. Basically free of charge, we can browse very well and very poorly designed Web pages. One scripting error or wrongly rendered page and an entire company's mission statement, "We produce quality software," could be a joke.

Marketing and product management constantly face the challenge of promoting a "value" with the message, but the competitor is just a mouse-click away. Picture a gourmet high-priced food store that wants to expand and sell groceries via the Internet. Customers visiting the actual store can smell the fresh products, touch and experience the inventory, and meet and chat with neighbors. It would take superior design skills to broadcast a similar experience over the Internet. Even with excellent products and services, the gourmet food store might lose business against discount supermarkets on the Internet, and a template approach would fail.

*Conclusion: Innovation is value.*

### **Managing Good Software Projects**

If you ask an author how long it will take to write a new bestseller, he most likely will say, "How do I know?" If you had asked me as a programmer how long it would take me to solve the programming issue described in the beginning of this article, I would have answered in the same way. How would I know that the solution would hit me after three days? IT project managers ask questions like the one above and expect an honest answer. We realize the difficult scope of this question if we ponder our own undefined question, "How much time will it take you to plan and manage this project?"

When assembling a car, we can ask an engineer, "How long will it take you to add the wheels to the car?" "Four-and-a-half minutes," he might respond. But then ask, "How long does it take you to engineer a brand new car from scratch?" and see what kind of response you get.

For example, the weather forecasters never guarantee any weather development. They predict and estimate the likeliness by watching weather patterns. The more local the source of the weather forecast, the better the predictions. In software engineering we ask for a guarantee because estimations and likeliness do not match the standard for "hard" numbers taught in business schools. And software projects are often run or sponsored by business units.

Good software is made by good teams, including the project manager. Often times, project managers are external to the team environment rather than part of it. Running a project by forecasts and interviewing project members is less predictable than reporting on the current state and the feel of the team by being inside. I've found that good software is usually produced by people who complement one another. A hundred Java developers aren't the right mix if none of them can write and manage requirement statements or has the right design skills.

I have seen a lot of project plans, many of them documented using a work-breakdown structure (WBS). Some of those plans were so detailed that actually doing the task was faster than the planning process. Project managers can monitor those milestones on paper and can execute the escalation routine if deadlines of activities are not met. However, at that point in the project it is too late, because the risk has already materialized into a problem. A WBS works well in manufacturing, where input and output parameters

are better understood. In software engineering, a plan has to be more flexible and more adaptive, for a too detailed WBS eventually fails.

In a similar approach, I have looked at many project status reports that indicated "on time" and "on budget," with "high morale" among the team members. Talking directly to the team members gave me the totally opposite picture. How successful do you think a soccer team would be if the coach didn't show up for training sessions and received the team's results and performance from the local newspaper?

*Conclusion: Forecast local developments.*

### **Media and Communication**

Before telephones and Web-casts existed, people needed to be in the same room or write letters to each other in order to communicate. Bringing people together at the same time, in one location, and writing, stamping, and delivering mail consumed quite some time. In other words, you needed to say something important and resources weren't to be wasted.

Today, with people and information at your fingertips, such a time commitment doesn't seem necessary. Electronic media even allows virtual meetings, sometimes with simple nicknames. This anonymous collaboration is very popular and successful for a variety of reasons. However, I personally believe electronic communication best complements conventional meetings held in person. The isolation caused by those technologies can create a distance between team members and the project.

When people physically meet, a team appears. Picture yourself among others in a roundtable meeting. Some participants don't have the time to actually make it to the meeting and instead decide to dial-in via phone conference. In almost all of those meetings I have been in, the people in the room interacted more, discussed, and shared laughs, whereas the participants on the phone seemed more like outsiders. Body language is a very important piece of human communication, but we shouldn't forget the pre- and post-meeting communication—exchanging handshakes, sharing personal stories in the elevator while going to the meeting, and mingling after the meeting when leaving the meeting room. Sound familiar?

Electronic communication is not limited to the professional life anymore. People may find their partner for life online, but they usually meet before they get married.

*Conclusion: Humans need human interaction.*

The scenarios in Part I have presented some challenges in software engineering in general, and they exist even if projects are not executed offshore. In Part II, I elaborate on these topics and discuss the challenges when projects go offshore.

### **About the Author**

Jochen "Joe" Krebs, <http://www.jochenkrebs.com>, is a method engineer within the Rational Brand for IBM. He develops content for the Rational Unified Process, OpenUP, and other agile software engineering processes. Prior to his current role he was responsible for successful enablement of Rational products and services for clients in the financial sector. Before joining IBM Rational he worked as an instructor and senior consultant with a focus on project management, requirements management, software engineering processes, and object-oriented technologies using Smalltalk and Java. He holds his MSc in computing for commerce and industry at the Open University.