Agile Offshore Outsourcing

Concepts and Practices for Flexible Integration of Offshore Development Services

Agile Business Conference 2006

Joachim Sauer
Agenda

- Challenges and common solutions of applying agile methods to offshore outsourcing
- Our approach: attention to established practices of software engineering, esp. architecture-centric development with components
- Our main empirical basis: case study in the mail order business
Our Background

Research-oriented practice and practice-oriented research

➔ C1 WPS GmbH
  • Founded in November 1999 as spin-off of the University of Hamburg
  • Part of C1 Group, based in Germany
  • IT consulting, training and development

➔ University of Hamburg, Dept. Informatics, Center for Architecture and Design of IT-Systems, Software Engineering Group
  • Led by Prof. Dr. Christiane Floyd and Prof. Dr.-Ing. Heinz Züllighoven

➔ Both:
  • Focus on object-oriented programming, frameworks, development processes and software architecture
  • Human-centered approach, emphasizing cooperation with users and evolutionary development strategies
  • Long agile tradition, mainly based on XP
  • Tools & materials approach (T&M approach) as primary software development method
Terms and Definitions

➡️ **Outsourcing:**
  • Transfer of business functions and structures to third party companies.

➡️ **Offshoring (= Offshore Outsourcing):**
  • Transfer of outsourcing services to low-wage countries, esp. for software development. Geographical and not contractual issues take centre stage, contrary to other variants of outsourcing.

➡️ **Types of offshoring (leitmotiv)**
  • Project support by offshore personnel in the customer's country (human resources assistance)
  • Complete project execution offshore, based on an exhaustive specification (offshore software factory)
  • Dual-shore development with close collaboration and common processes and artefacts (one team)
Offshoring Promises

- Reduced costs
- Access to a larger development workforce
- 24/7-development
- Concentration on core competences
- Faster access to special know-how
- Usage of modern technologies without own investment
- Higher product quality
- Risk sharing with the offshore contractor
- ...
Common Problems with Offshoring

- Customer management
- Communication requirements
- Legal and contractual implications
- Costs of travel and infrastructure
- Intellectual property rights
- Culture
- Quality assurance
- ...

Agile Offshore Outsourcing
Joachim Sauer, 2006
Why Use Agile Practices?

Offshoring (cost savings) + agility (flexibility) = successful projects!? 

- Waterfall development processes are not desirable in general.
- In offshoring settings, the big bang could be even worse.
- Agile practices seem preferable:
  - Iterative, evolutionary proceeding with prototypes
  - Communication and feedback on all levels

[Manifesto for Agile Software Development](http://agilemanifesto.org/)

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.
Agile Offshoring Terms and Definitions

- Agile Offshore Outsourcing (AOO)
  - Application of agile practices to offshoring projects.

In contrast to

- Agile Global Software Development
  - Teams may be of equal status

- Distributed Extreme Programming
  - Application of extreme programming as agile process model
  - Not necessarily in offshoring settings
Results from a recent analysis of published experiences:

- **Agile Offshore Outsourcing** is used successfully in some projects.

- **Main problems and restrictions:**
  - **Collaboration:** spreading of informal news, lower team spirit, coordination problems, creation of common knowledge and a shared vision, cultural differences
  - **Development issues:** inadequate infrastructure, difficult collective code ownership, integration and configuration headaches, quite different design and architecture skills of developers, training of the offshore developers, quality assurance
  - **Requirements engineering and customer involvement:** customer on-site mostly impossible, language barriers, long feedback intervals, establishing credibility with the customer
  - **Project management:** estimation of global and task progress, knowledge transfer, supervision and controlling

- **Basically, it is about Communication.**

[Sauer, 2006]
Tackling the Communication Challenge

Postulations

- Speak the same language!
- Make tasks explicit and clear!
- Define clear interfaces (software and human)!
- Enable multiple and rich communication channels!

What has been tried?

- Video- and telephone conferences, electronic project rooms, remote meeting software, wikis, distributed pair programming
- Frequent visits/travels
- Shared kickoff-meetings
- “Ambassadors”, “Rotating Gurus”
- Common version control systems
- “Proxy Customers”
- Video conferences with the customer
- …
Our Research Questions

Agile Offshore Outsourcing may work, as experience has shown, but it is expensive and requires a complex and time-consuming setup.

This prevents its application to smaller and shorter projects.

Our Research Questions:

- How can agile offshoring be used in smaller projects?
- How to integrate offshore software development tasks in conventional development projects?
Our Research Area

Bertrand Meyer states:
“Along with education, software engineering research should take account of outsourcing. Existing software engineering principles and tools can, as noted above, help offshore projects, but there’s far more to do.”

We follow this advice and concentrate on **software architecture and components** in Agile Offshore Outsourcing.

- Architecture-centric development
- Component-based development

Also important, but not in our focus

- Roles in the process
- Efficiency calculations
- Selection of the offshore provider
- Contract issues

[Meyer, 2006]
“The **software architecture** of a program or computing system is the structure or structures of the system, which comprise **software elements**, the **externally visible properties** of those elements, and the **relationships** among them.”

[Bass, Clements, Kazman, 2003]

**Architecture-based development** “differs from traditional development in that it concentrates on **driving design and maintenance from the perspective of a software architecture**. The motivation for this change of focus is that a software architecture is the placeholder for system qualities such as performance, modifiability, security, and reliability. The architecture not only allows designers to **maintain intellectual control** over a large, complex system but also **affects the development process itself**, suggesting (even dictating) the assignment of work to teams, integration plans, testing plans, configuration management, and documentation. In short, the architecture is a **blueprint** for all activities in the software development life-cycle.”

[Bass, Kazman, 1999]
Benefits of Architecture-Centric Development

➔ **Comprehension and communication**
  • Common language for all stakeholders
  • Concrete artefact for precise discussions and arrangements

➔ **Record of experiences**
  • Reuse of knowledge of similar projects
  • Represents technical and business-relevant decisions
  • Documentation of design rationale

➔ **Construction**
  • Blueprint for developers
  • Verifiable architecture rules ensure consistency of the implementation

➔ **Project documentation**
  • Abstract description of the application
  • Aids in understanding the source code

➔ **Evolution**
  • The architecture anticipates required future changes to the application
  • It enables variability on determined points

➔ **Management**
  • The architecture is the basis for the project organisation
  • It aids in assigning tasks to concurrently working teams
  • It reflects the progress of the project
Case Study in the Mail Order Business

Goal:

- Evaluation of architecture-centric development based on the established tools & materials approach

- Development of a commercial prototype for the modernisation of an order entry and customer information system

- Dual-shore offshoring with German business analysts, developers and software architect and Indian developers

- Agile process model based on Extreme Programming

- Focus on architecture definition, evolution and inspection

- Architecture documents from the case study:
  - Model architecture
  - Highlevel view
  - Implementation view
"A model architecture abstracts from the characteristic features of a set of similar software architectures. It defines the kind of elements used, their connection and the rules for their combination. In addition, a model architecture includes criteria for the composition of elements into modeling and construction units, and guidelines for the design and quality of a specific architecture."

- Functional and logical reference architecture
- Adaptable for particular application domains as domain-specific software architectures
- Should be based on suitable metaphors
- Can be supported by frameworks and automated architecture rule checkers

Framework for the tools & materials approach (open source, www.jwam.de)

[Züllighoven, 2004]
Different Architecture Views: From High Level View ...

**Tools**
- Main tool
- Customer tool
- Order tool
- Terms of delivery tool
- Product details tool
- Order item tool

**Materials**
- Customer
- Article
- Order
- Terms of delivery
- List of orders
- Order item

**Domain-values**
- Article number
- Customer number
- E-mail
- Date
- Title
- Quantity
- Rate

**Services Automatons**
- Persistence service
- Persistence service for articles
- Persistence service for orders
- Persistence service for customers

**External Systems**
- DB2
- File
- MySQL
- File
Due to copyright reasons, a similar but different project's architecture is shown.

In this diagram, only call relations are shown for clarity.

1 Start classes
2 Main tool
3 Sub tools
4 Automatons
5 Services
6 DB service
7 Materials
8 Domainvalues
9 JWAM
Component-Based Development

- How can tasks be assigned on the basis of a common architecture?
- XP’s story and task cards are too coarse and ambivalent
  - Cannot be implemented without further inquiries
  - Differences of interpretation can arise

- The solution: Components
- User requirements are captured and broken down in components based on the architecture by on-site analysts and developers.
- Components are then implemented on-site or offshore ...
- … and checked and integrated on-site.

- Components can be of different sizes and can be ordered hierarchically. So gradually more and more tasks can be implemented offshore.
Example Component Description

- **Related story:**
  - Tools for terms of delivery

- **Type:**
  - Sub tool

- **Task:**
  - Implementation

- **Description:**
  - Implement a tool for the input of account information that is integrated into the tool for terms of delivery and automatically shown when the customer chooses to pay by direct debit. The following fields and check routines have to be implemented (see sketch for layout):
    - account number (German: Kontonummer), based on account number domain value

- **Tests:**
  - [...] 

- **Effort points:**
  - 2

- **Recorded by:**
  - Joachim Sauer, 11.03.2005

- **Status:**
  - approved, 01.04.2005

During the actual project, a less formal description was used. The structure shown here was developed as a result of the case study.
Accompanying Practices

The following practices are valuable extensions:

- **Coding and documentation conventions (style guide, naming rules, ...)**
- **Test-first development**
  - Automated unit tests with Junit
  - Automated acceptance tests with FIT/Fitnesse
- **Quality assurance with automated tools**
  - Architecture validation
  - Check for cycles, interface violations, ...
  - Metrics (big classes, long methods, ...)
Findings and Recommendations

- Agile Offshore Outsourcing may not be as agile as you want it to be.
- Communication is the big issue. You will need more documents.
- Architecture-centric development seems to be a step ahead.
  - Good model architectures, metaphors and frameworks help to educate developers and ease implementation.
  - The need for communication decreases.
    - Communication is more goal-oriented.
    - In the case study, only project leads (on-site and offshore) communicated directly.
    - Only business analysts communicated with the customer.
  - Component-based development enables flexible integration of offshore development services. Small packages can be developed offshore under clear rules.
  - Training of the offshore developers is very important. Sufficient time should be spend on it as the learning curve can be quite steep (agile practices, working with architecture views).
  - Quality assurance is also very important. Automated checking of architecture rules and metrics prevented architecture violations.
  - Test-driven development was difficult in the case study as the offshore developers had no experience with it and its mastering is challenging.
Bibliography and Links


- C1 WPS: http://www.c1-wps.com
- Software Engineering Group, University of Hamburg: http://swt-www.informatik.uni-hamburg.de/