Overseas Development for a Major U.S. eCommerce Website

Jiang Wu, PhD.
eHealth Inc.
1-650-210-3122
jiang.wu@ehealth.com

Sheldon Wang, PhD.
eHealth Inc.
1-650-210-3188
sheldon.wang@ehealth.com

Christine Chau
eHealth Inc.
1-650-210-3164
christine.chau@ehealth.com

Lei Zeng
eHealth-China Inc.
86-592-539-4694
lei.zeng@ehealth-china.com

Jinsong Lin
eHealth-China Inc.
86-592-539-4695
jinsong.lin@ehealth-china.com

ABSTRACT
In this paper, we describe our experience in establishing a software development center in China with the purpose of supporting a major U.S. eCommerce website. We have established a set of development processes that fit our business needs to develop a large number of relatively small projects and release them in very short intervals. Our processes allow us to accurately monitor the project status, schedule resources, predict the delivery and assess the productivity. We believe only with such systems in place that businesses can ensure successful overseas software operations.

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]: Management – *Cost estimation, Life cycle, Productivity, Programming teams, Software configuration management, Software process models, Software quality assurance, Time estimation*

General Terms
Management, Measurement, Economics.

Keywords
Development process, Offshore operations, Productivity

1. INTRODUCTION
In the past few years, software companies in the West have been shifting their development effort into Asian countries. The benefits they seek include access to a large pool of well-educated technical talents, reduced labor cost and 24 hour operations. As the result, businesses are forced to operate multiple development centers across countries. Typically home country has the emphasis of business & product management, while offshore sites focus on product development. Such remote or multiple development sites post specific challenges for companies. Examples are communication barriers including time and language differences, lack of formal software development experience in the talent pool, differences in culture and management styles. There is no lack of examples where companies are ill prepared in starting such ventures. The success of a company's offshore operation depends on how they leverage the benefits and overcome the challenges.

eHealth Inc started its Chinese operations (eHealth-China) four years ago in the city of Xiamen. eHealth is the largest online health insurance broker in the United States. Its website (www.ehealthinsurance.com) sells health insurances products to individuals and small businesses from all major US insurance companies across 50 States. Similar to other consumer facing portals, the site requires frequent content updates and feature enhancements to keep up with customer needs. Our eHealth-China operation is responsible for a major portion of these short-duration, large-quantity types of projects with tight weekly release schedules. From the get-go, we have determined that establishing sound development processes that fit with our specific business needs is the key to address the challenges mentioned above and therefore to ensure the success of our China operation. In the following sections, we will summarize our development environment, our project tracking methodologies and the tools we have developed to facilitate our process. We will also discuss a preliminary finding in how our Chinese operation faired in development productivity.

ICSE'06, May 20-28, 2006, Shanghai, China.
ACM 1-59593-085-X/06/0005.

2. DEVELOPMENT PROCESSES FOR DIFFERENT PROJECT TYPES
Investments in process improvement have consistently yield superior products and significant returns for businesses [1][2]. We believe this is especially important in the offshore and multi-site development environments. With different development methodologies available [3][4][5], choosing the proper ones and tailoring them to the specific project needs are essential. Here is how we matched up the process with each type of project we perform.

2.1 Small Maintenance Projects

These are projects that take an average skilled engineer between one to ten days to develop. Because of their limited scope and similarity to other implemented projects, we use a rather strict waterfall process for handling them. We strive for precise time estimates and minimum defect rates as the project goes through a linear specification, implementation, testing, and release cycle. Examples of such projects include implementing a new insurance carrier on our site, adding support for a new affiliate, or making a small feature change. Because of the large volumes and short durations of such projects, traditional project management tools such as Microsoft Project become cumbersome and inefficient. We have designed a streamlined workflow process and associated tools to manage the life cycle of such projects, which will be described in the later sections.

2.2 Medium-size Enhancement Projects

These are projects that last two to six weeks with reasonably clear scope and objectives. Examples of such projects are enhancements to a subsection of our website or data interface with insurance carriers. In this category, we have mixed success using a straight waterfall process. We have learned to build in cushions for limited iterative process to take place.

2.3 Major Changes or Infrastructure Projects

These are large projects that are at least several months in duration involving multiple engineers. Their nature and scope tend to be vague and unpredictable initially. Often times, the project is divided into multiple phases and we can only determine the next step after the previous step has reached certain plateau. We have found agile development processes such as Extreme Programming (XP) suite well with this type of projects [3][4][5]. Examples of such projects are major site flow updates or underline architecture changes to support online insurance application process. Because it is difficult to predict exactly what customers or market will demand from day one in such projects, we normally take a spiral or incremental process in an effort to elicit a lot of feedback in earlier phases. This way, we can learn the needs of the users better and react accordingly.

3. DEVELOPMENT AND DEPLOYMENT ENVIRONMENTS

To support development of the three types of projects through our China operation, we need an infrastructure that provides the flexibility to coordinate development efforts between our China and US teams. Figure 1 depicts how we set up our environments to develop products in our Chinese operation and propagate them through to production release on US side. The physical setup is based on using a Source Code Control System with multiple code branches and a series of hardware environments (web, application and database servers) that can host our software for each one of the code branches.

3.1 Development Environments

These environments, shown in Figure 1, are composed of multiple code branches supporting concurrent development of different projects. The hosting servers are located in our China location. Each development code branch can be deployed to the servers to provide early look of the software being developed. They are also used for developer unit testing and initial functional QA testing of features. Each development branch is regularly synchronized with the released code (described below) to ensure it is up-to-date with all released features.

3.2 Integration Environment

Projects ready for final QA testing are integrated from their perspective development branches into the integration branch according to release schedules. The integrated code are built multiple times daily and delivered to QA servers either on China or US location for final QA and regression testing. As part of the integration process, code conflicts between different projects are identified and resolved.

3.3 Staging and Production Environments

Code for projects ready to release is promoted to the staging branch. The build is delivered to the Staging servers for final QA verification and regression testing. At last, the verified release is built on the release branch and delivered to the production site. Both staging and production servers are located on the US side.

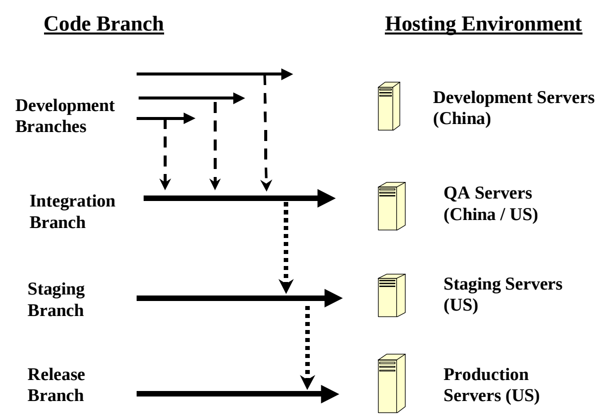

Figure 1. Code propagating through development and deployment environments

4. PROJECT CONTROL AND TRACKING

The design of environments above allows separation of different development activities and provides the framework for project tracking and monitoring at difference stages. As mentioned before, due to large volumes of small projects, it is impractical to use traditional project management tools for a majority of our projects. We have developed specific control mechanisms and tracking tools that are volume-oriented. Our processes facilitate individuals to move projects through the pipe quickly with minimum overhead when interacting with the project management tools. The tools then aggregate individual's inputs into project views that help managers to identify bottlenecks and move projects forward.

4.1 Project Control

A project must be initiated by a System Change Request, which is entered into a project tracking system. The project is moved through a life cycle of status changes from "New Project" to "Released". As illustrated in Figure 2, the project statuses are designed to match closely with the project promoting through the development environments. At each stage, a control point is put in place so specific criteria are checked off by the appropriate personnel before moving forward.

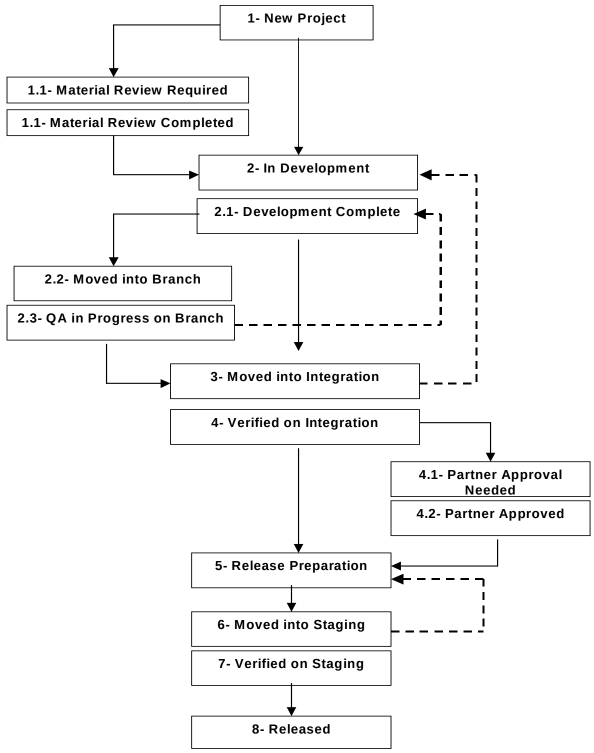

Figure 2. Project status flow diagram

Level-1 statuses (1 - 1.2) relate to project requirement reviews and engineering acceptance. With the exception of very simple projects, the project does not get moved to level 2 until development team sets status to "Materials Review Completed". Level-2 statuses (2 - 2.3) corresponds to the development phase conducted on the develop branch and development servers. For medium sized projects or those with major changes, component testing must be conducted and passed on development servers before the project is allowed to move into the level-3 status, which is "Moved into Integration".

For each subsequent advance of status level, QA team must first verify the project with required testing. In addition, system regression testing and load testing are always conducted on the Integration and Staging environments. The dotted arrows in Figure 2 represent the status change from a higher-level back to a lower level. This happens when the project has failed QA testing and is sent back for rework. Similarly, those projects that failed final regression testing are sent back to release preparation for repackaging. Such check and balance ensures project quality as the code moves closer to production.

4.2 Project Management Tools

We have developed and implemented a set of tools to manage our workflow according to the project status flow diagram in Figure 2. At any particular juncture, the project is assigned to an owner. As the current owner completes the required task, she uses the project-tracking tool to move the project into the next status dictated by the status flow diagram and to assign the project to the new owner appropriate for that status. Required user interaction with the tool is normally just a few clicks. The tool automatically calculates time and duration of the performed activities between status changes. This information becomes the basis for calculating various project metrics.

For project managers, the aggregated view of all on-going projects of interest in each phase of the life cycle is particularly useful. As shown in Figure 3, the top section is a list of projects ready to be released with the next release build. The bottom section lists projects currently in development. The middle sections are projects with statuses in between. The columns display the key information regarding each project, such as current status, current owner, project creation date, duration, targets, and other linked or dependent projects. When a project misses the expected schedule, the manager will be alerted in this view. The manager can also drill down to the project details.

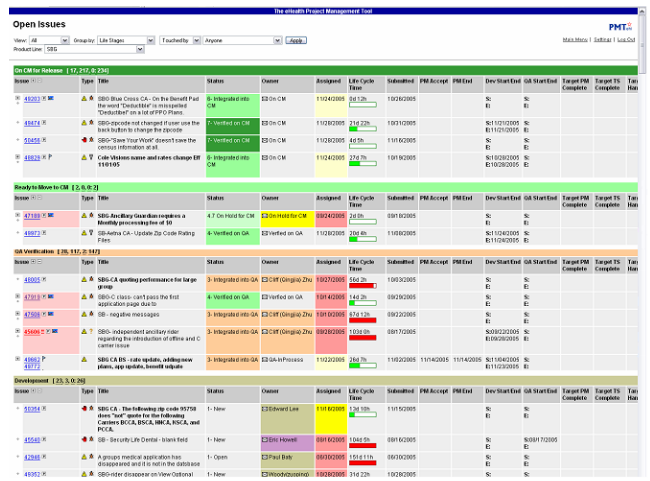

Figure 3. Project management tool that displays all active projects in different statuses

5. PRODUCTIVITY MEASUREMENT

Based on the project tracking in place, we have devised a set of performance measurements for our team and individuals. These measurements range from overall product output, defect rate found in each stage of development, to estimated and actual times taken for each project. One measurement of particular interest is for tracking individual and team productivity. We have designed a simple and informative indicator, which gives instant feedback to our China & US managers on individual and team performance. Such feedback enables managers to take immediate corrective measures to resolve project and personnel issues, and ensure timely delivery of products.

5.1 Design of Productivity Measurement

For each project that goes through our development pipeline, we define a Standard-Effort (SE) in terms of number of working days needed to complete the project by an average-skilled engineer. An engineer who finishes the project in less than the Standard-Effort is considered to have above average productivity. SE day is our unit of measure for individual and team productivity. Without counting other overhead, we would expect an average engineer to produce five SE days' worth of work in a week of five working days. As such, SE Days is a measure of work produced, not a measure of actual time spent.

Based on the above concept, we have set up a procedure to estimate the Standard-Effort for each type of project. And since a large number of our projects falls into small to medium sized projects, we are able to rather accurately estimate a standard effort for these projects. In most cases, we are using the counting of "feature" points to establish the standard effort. For example, to implement a new insurance company's products on our website, we estimate the complexity of the insurance rating system based on the number of different insurance plans offered, number of parameters affecting the rate quoting, and number of geographic regions covered. We then establish different SE days for projects that vary in these parameters.

We track productivity on weekly basis. An individual's productivity is calculated by adding up the SE days associated with the projects completed by the individual in that week (pro-rated by the percentage of project completed.) A team's productivity is the average of all members' productivity.

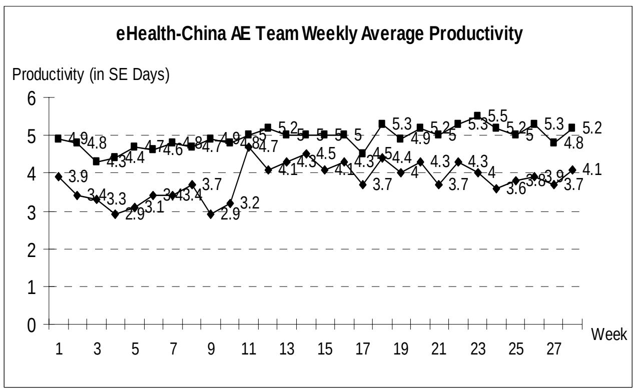

Figure 4. Weekly team productivity chart

Figure 4 shows the productivity chart for a typical eHealth-China development team in a 30-week duration. The lower series, project SE Days, tracks the weekly SE days directly related to developing projects. The upper series, total SE Days, also includes all other activities performed by the engineers such as team meeting, technical training, and other overhead. Understandably, we would like the project SE days to be as high as possible because it relates directly to the products delivered. We also would like to keep the gap between the product SE days and total SE days at a reasonable level. Too wide of a gap usually means too much overhead. Too narrow of a gap may indicate the team is overwhelmed with projects or may not have invested enough for staff improvement. With such measurements in place, we can analyze the trend and make comparisons historically across teams or individuals. This in turn allows us to develop best practices and evaluate effectiveness of process changes based on facts.

5.2 Comparison of China-US Engineering Productivity

Since both of our US and China development teams handle similar projects (US teams tend to handle more larger size projects), we are able to make some preliminary comparisons of productivity between the two locations.

Table 1 shows for all small size and a good portion of medium size projects, our Chinese team yields the same productivity as the US team. The metric is more accurate with smaller projects, but becomes harder to estimate as project size gets larger and project becomes iterative in nature. Our observation has been that with these larger projects, the level of interaction required between business units (located in US) and the development teams have cut into the productivity of our Chinese team. These are the issues we will be addressing in our next phase of process refinement.

Table 1. Chinese engineering productivity relative to the baseline of 5 SE Days performed by the US engineering team

	Small Projects	Medium Projects	Large Projects
US	5	5	5
China	5	7.5	10-15

6. CONCLUSIONS

eCommerce website development and maintenance using overseas development centers are demanding and challenging operations. Sound software engineering processes and tools, if selected properly and applied to suit the organization's specific environment, can overcome time, distance, and language barriers to ensure the success of such operation. Our four-year experience in China has shown that with the right processes and tracking, our China operation can achieve near one-to-one productivity compared to our US operation for most development projects.

7. ACKNOWLEDGMENTS

The authors thank many of our colleagues at eHealth and eHealth-China who contributed to establishing the development processes described in this paper.

8. AUTHORS' ADDRESSES

The address of eHealth Inc is 440 E. Middlefield Rd. Mountain View, CA 94043, USA

The address of eHealth-China Inc is 6F, Software Technology Service Bld. Software Park, Xiamen , PRC

9. REFERENCES

[1] Humphrey, W. *Managing the Software Process.* Addison-Wesley, Reading, MA, 1990.

[2] Carnegie Mellon Software Engineering Institute. *Demonstrating the Impact and Benefits of CMMI: An Update and Preliminary Report,* www.sei.cmu.edu/publications/documents/03.reports/03sr009.html, Oct. 2003.

[3] Beck K., et al. *Manifesto for Agile Software Development,* www.agilemanifesto.org, Feb. 2003.

[4] Beck, K. *Extreme Programming Explained,* Addison-Wesley, Reading, MA, 2000.

[5] Fairley, R. E., and Willshire, M. J. Iterative Rework:The Good, the Bad, and the Ugly, *Computer, 38,9(Sep. 2005), 34-41*