Agile Software Processes for the 24-Hour Knowledge Factory Environment

Nathan Denny, University of Arizona, USA
Igor Crk, University of Arizona, USA
Ravi Sheshu, University of Arizona, USA

ABSTRACT

The growing adoption of outsourcing and offshoring concepts is presenting new opportunities for distributed software development. Inspired by the paradigm of round-the-clock manufacturing, the concept of the 24-hour knowledge factory (24HrKF) attempts to make similar transformations in the arena of IS: specifically to transform the production of software and allied intangibles to benefit from the notion of continuous development by establishing multiple collaborating sites at strategically selected locations around the globe. As the sun sets on one site, it rises on another with the day’s work being handed off from the closing site to the opening site. In order to enable such hand offs to occur in an effective manner, new agile and distributed software processes are needed, as delineated in this article.

Keywords: 24-hour knowledge factory; agile software processes; offshoring; outsourcing; scrum

INTRODUCTION

The industrial revolution led to the concepts of assembly lines, shifts for factory workers, and round-the-clock manufacturing. Advances in IS now enable us to envision the non-stop creation of new intellectual property using shifts of workers located in different countries. More specifically, a 24-hour knowledge factory (24HrKF) is envisioned as an enterprise composed of three or more sites distributed around the globe in a manner that at least one site is operational at any point of time (Gupta & Seshasai, 2004). As the sun sets on one site, it rises on another with the work in progress being handed off from the closing site to the opening site.

Earlier articles on the 24-HrKF have broadly classified professional work as being ill-structured, semi-structured, or totally structured (Gupta & Seshasai, 2007; Seshasai & Gupta, 2007). CEOs, presidents, and other heads of organizations usually deal with ill-structured work, the pattern of which cannot be predicted in advance. This type of work cannot be easily decomposed into subtasks that a shadow-CEO or a partner-CEO, located in a different part of...
the world, can complete during the next shift. At the other end, work in industries like call centers is well-structured and can be readily picked up by a colleague coming in to work for the next shift and located in another part of the world. In between these two extremes, there are many examples of semi-structured work where the overall endeavor can be broken into subtasks and a person in the next shift can continue to work on the incomplete task from the previous shift. Software development, in specific, and many IP-based industries in general, fall into this intermediate category.

In the conventional model of software development, tasks are usually divided on a horizontal basis among teams in different geographies; that is, one team in one part of geography developing one module, and another team in other part of the geography developing the other module. In this model, if one part of the project gets delayed, the developers working on this part end up having to devote extra hours, typically by working late into the night. Colleagues working on other parts of the project are unable to render help, because they are only familiar with their respective parts of the project. In the 24HrKF paradigm, this problem can be overcome by the vertical division of tasks between developers in different geographies. The latter goal can be met only with sustained research of relevant underlying issues, and the development of new agile and distributed software processes that are specially geared for use in the 24HrKF environment. For example, new methodologies are needed to streamline the hand-off between the developer in one shift and the developer in the next shift in order to communicate details of the task in progress, as well as pending issues, in a very rapid and efficient manner in order to reduce the time spent in the hand-off to the minimum possible. Further, new task allocation processes need to be developed to address the reality that the developers will possess different skill sets and dissimilar levels of productivity. Such issues are being researched by a team involving individuals from academia and industry. The research group at the University of Arizona is working closely with colleagues in universities in Australia and Poland under the aegis of a collaborative agreement signed in 2005.

The research team at the University of Arizona has designed a process “CPro” that addresses several of the operational issues related to the 24HrKF environment. The core of CPro is a model of cooperative work called the *composite persona* (CP). A CP is a highly cohesive team of developers that are distributed across the globe. When the problem is decomposed, it is decomposed horizontally as is conventional. However, subcomponent development is now assigned to a CP rather than an individual. The members of the CP work on the vertically decomposed subcomponent in series, each successive shift improving upon the work of the previous shift. Based on the CPro process, a tool, Multimind, has been designed, developed, implemented, and tested.

**DISTRIBUTED AND AGILE SOFTWARE DEVELOPMENT**

Agile processes are relatively new to the field of software development and have recently accumulated both popularity among developers and a portfolio of significant, successful applications. “Agilists,” as practitioners of agile software processes call themselves, are loosely united under the statements of the agile manifesto (Fowler & Highsmith, 2001). This manifesto asks subscribers to place emphasis on processes that are practical, adaptable, and produce artifacts that have value for stakeholders.

Agile software processes are a class of processes rather than a single, specific technique. While agile processes are sometimes criticized as being reactionary to current software engineering dogma, they are not necessarily orthogonal to traditional “tayloristic” software development. More specifically, traditional software engineering endeavors to establish highly defined, functionally specialized roles and encode almost all knowledge about the software under development into explicit, formal documents. In contrast, agile processes, although they are numerous and differ between
13 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the product's webpage:

www.igi-global.com/article/agile-software-processes-hour-knowledge/3692?camid=4v1


www.igi-global.com/e-resources/library-recommendation/?id=2

Related Content

Disturbing Realities Concerning Data Policies in Organizations
www.igi-global.com/article/disturbing-realities-concerning-data-policies/50929?camid=4v1a

A Path Analytic Study of the Antecedents of Organizational Commitment of IS Managers
www.igi-global.com/article/path-analytic-study-antecedents-organizational/1202?camid=4v1a

Information Systems and Technology Outsourcing: Case Lessons from “TravelTrack”
Jeremy Rose and Ray Hackney (2001). Pitfalls and Triumphs of Information Technology Management (pp. 141-152).
www.igi-global.com/chapter/information-systems-technology-outsourcing/54280?camid=4v1a