A Quality Perspective on the Menlo Software Factory™ Software Development Methodology

12-Jan-05
Carol Perletz
Senior Manager of Software Quality Assurance

NOKIA

Agenda

1. Introduction
2. The Software Crisis
3. Menlo Innovations
4. Why the Menlo Software Factory Approach?
5. Education Crisis
6. Promising Educational Reforms
7. Parallels Between Software and Education Crises
8. Conclusions From My Personal Experiences
9. References

NOKIA

Introduction

NOKIA

Introduction

- My background
  - Went through American middle class public school system during mid 1950s through 1960s
  - 1972, 1975: BA Ed., MS Ed.
  - 1969, 1971 – 1974: worked for an R&D defense contractor
  - 1974 – 1977: taught in the largest Cleveland elementary school
  - 1977 – 1979, 1984 – present: worked in software companies
- My experiences
  - Teaching and professional training, curricula development
  - Product and test programming
  - Management of technical professionals, business unit, startup VP
  - Set up, managed, individual contributor of SQA groups
  - Organizational and process development in schools and companies
  - Product management, user interface design, wrote user documentation, online help development

NOKIA

Introduction

- Why am I doing this presentation?
  - In May 2004, my Nokia group was looking into setting up an Innovation Center
  - I had done a lot of investigation into software development methodologies, participated in a SCRUM project, and worked with others to write waterfall, modified waterfall, spiral processes, and an integration process for an iterative model since 1984
  - In late May I was sent a link to Richard Sheridan's Menlo Innovations Web site: http://www.menloinstitute.com/index.htm
  - I was so impressed, I tried to set up a discussion group with other professionals regarding Sheridan's software factory
  - Steve Rakitin asked me why I was intrigued since he questions the "hype of agile methods"
  - My answer: the best teaching methods I found along with the elements of software product successes I experienced included many of the main tenets of Sheridan's Menlo Software Factory™

The Software Crisis

The Software Crisis

- Current crisis in software development
  - Software projects are cancelled
  - Cost overruns in billions of dollars for late projects
  - 80% of project dollars go towards fixing defects
- Recommended solutions:
  - W. Edwards Deming - 14 points for transforming management
  - Joseph M. Juran - trilogy of managerial processes for quality: quality planning, quality control, and quality improvement
  - Watts S. Humphrey – company (CMM), team (TSP), and personal (PSP) processes for managing software development
  - Quality Function Deployment (QFD) – structured method for managing customer requirements
  - ISO 9000-9004, 2000, Malcolm Baldrige Award: quality certifications and awards for companies

The Software Crisis (con't)

- Recommended solutions (continued):
  - IEEE Software Engineering Body of Knowledge
  - Carnegie-Melon University Software Engineering Institute (SEI)
  - Project Management Institute (PMI)
  - Identification of software development life cycle models:
    - Waterfall
    - Spiral
    - Cleanroom
    - Spiral
    - Iterative
    - Incremental
  - Software methodologies:
    - Rational Unified Process (RUP)
    - Extreme Programming
    - Just-in-time (JIT)
    - SCRUM ("flavor" of agile modeling)
    - Agile modeling
  - Definition of standards, professional certification programs, templates, & forms to guide project artifacts, coding, software and test engineering

Menlo Innovations

Menlo Innovations

- 1876 - Thomas A. Edison set up an Invention Factory in Menlo Park NJ because: “There is always a way to do things better, find it.”
  - 200 workers with a wide variety of specialized skills
  - Worked together in a 1-room, open, collaborative environment
  - Worked iteratively and incrementally
  - Produced a minor invention every 10 days
  - Produced a major invention every 6 months
  - All products had to be valuable and marketable
- Richard Sheridan’s response to the software crisis: “business-centered, creative, and adaptive teams use agile processes to produce exceptional quality software while embracing change. Learn how to embrace change and deliver software to meet business needs when the business needs it.”
- Richard Sheridan founded the Menlo Innovations Institute, and created the Menlo Software Factory™, modeled after Edison’s Invention Factory

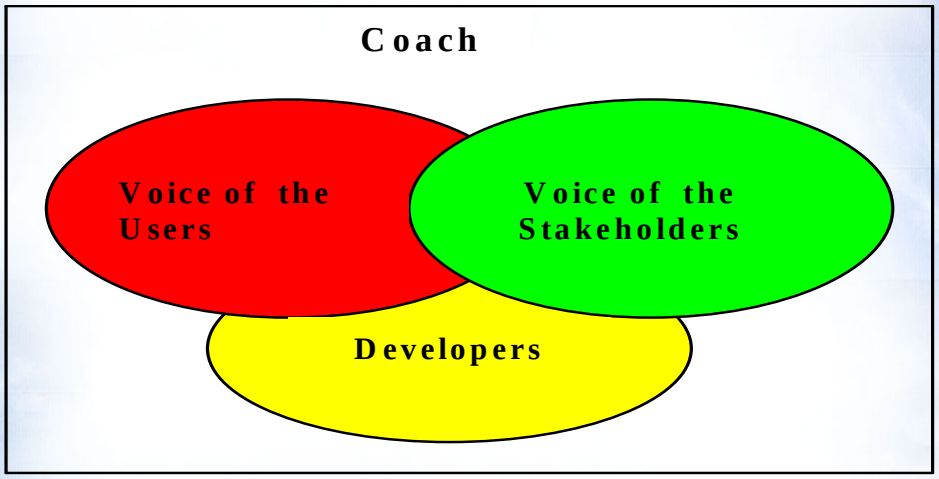
Menlo Innovations (con't)

- Characteristics of the Menlo Software Factory™:
  - Delivers features in any order
  - Releases features at any time
  - Business values drive software development, not technology
  - Clear mechanism to understand the development process
  - The business has a clear mechanism for driving development
  - Quality is intrinsic to the process, not inspected in
  - Knowledge is continually communicated
- Why use a factory/manufacturing process model?
  - In a factory, optimization is done around the entire production facility rather than the individual worker
  - The other popular model is the "craftsman model", first used during the Middle Ages, where optimization is done around the craftsman, journeyman, or expert

Menlo Innovations (con't)

- Why have an agile software development team?
  - An agile team can add features in the order that makes sense to the business, any order
  - An agile team can present frequent iterations for demos, trials, betas at any time
- How is the Menlo Software Factory™ organized & why?
  - In order to understand the software development process, a project team has to have "Profound Knowledge" (W. Edwards Deming)
  - Profound Knowledge – how people work, learn, interact, grow, succeed and fail, including an understanding and application of psychology
  - A software factory is made up of 4 teams, each with one voice, and each made up of highly skilled, specialized, dedicated members (one project, one iteration at a time)

Menlo Software Factory™ Teams



NOKIA

The 4 Teams

Coach – make the voices work together

Voice of the Users – represents the user goals and abilities

Voice of the Stakeholders – represents the business goals

Developers – write the software

NOKIA

Coach Role

- Coach – responsible for the process as a whole
  - Characteristics:
    - Voracious reader – recommended reading list
    - Most knowledgeable member of the team
    - Experienced project manager
    - Presents a formal methodology
    - Leads small project milestones
  - Team members:
    - Methodologist
    - QA manager
    - Project manager
  - Responsibilities
    - Effective communication facilitating:
      - Interactive Planning Sessions
      - Daily Stand-up Meetings
      - Show and Tell Presentations
    - Promotes stability in midst of change (developers not interrupted during an iteration)
    - The facility
    - Improving the process
    - Assigning developer pairs

Voice of the Stakeholder Role

- Voice of the Stakeholder - project ownership for building a system that is profitable for the company
  - Virtual team
  - Responsibilities
    - Stakeholder identification
    - Creation of the business case
    - Prioritization of tasks (understands development estimates)
    - Project plan development on behalf of the stakeholders, including:
      - Resources needed
      - Scope of the iterations and project as a whole
      - Time to be spent on the project (number of iterations)
    - Decides when the software is to be released – when there's enough functionality to be of value to the business and the users

Voice of the Users Role

- Voice of the Users – practice High-Tech Anthropology™
  - Characteristics:
    - Specialize in understanding corporate cultures
    - Improve the process of designing, creating, introducing, & adopting new technology
    - Use a cultural context to understand the users real needs
    - Empathetic, sympathetic, compassionate individuals
    - Work closely with users and engineers
  - Responsibilities:
    - Express user needs in a system design
    - Elicit requirements
    - Design usable systems
    - Create user interface designs and give detailed instructions to the Development Team
  - Multiple member roles included:
    - Business process analysts
      - Accurately reflect business concerns in business requirements
      - Know where the system will be deployed
      - Perform cost-benefit analyses
    - User interface and usability specialists
      - Develop simple and practical UI designs
      - Understand human factors and cognitive psychology
      - Create personas for each user type

Core Practices of the Menlo Software Factory™

- Open and collaborative production work environment
- All 4 teams run simultaneously throughout a project
- A completed project is delivered every 2 weeks as an iteration
- The software is developed iteratively and incrementally

Factory Layout

- No cubes
- Flexible floor plan in one room
- Long folding tables clustered in groups of 3 or 4 with chairs on rollers
- One computer for 2 developers
- Busy, alive, energetic, noisy
- Walls are decorated with status displays, software metaphors, key practices

Whole Group Activities

Interactive Planning Sessions

- Held the day before each iteration begins
- The Voice of the User Team writes out the planning tasks
- The Development Team estimates the tasks
- The Voice of the Stakeholders prioritizes the tasks
- The Coach assigns the tasks to the developer pairs

Whole Group Activities (con't)

Daily Stand-up Meetings

- 15 minutes max
- All team members stand in a circle
- Each team member talks once, one at a time
- Each team member answers 3 questions:
  - What have you accomplished today?
  - What are you working on?
  - What questions do you have?
- Questions are answered one-on-one outside the meeting

Whole Group Activities (con't)

Show & Tell Presentation

- Held at the end of each iteration
- Developer pairs install their software onto a production system
- Developers demonstrate the working functionality
- All teams and executives, sponsors, and users attend

Project "Artifacts"

Project Plan

- Created by all project team members
- Allows for "what if" scenarios to vary resources, scope, and schedule
- Consists of rows of sheets of paper covered with tasks
  - Each row is the size of the development team working on the iteration
  - Multiple rows indicate the number of iterations
  - Folded task cards are proportional to the developer's work estimate
- Tasks outside the rows are outside the project

Project "Artifacts" (con't)

Status Board

- Bulletin board hung in the factory for all to see
- Duplicate of the tasks in the project plan for the current iteration only
- Started tasks display a yellow dot
- Completed tasks display a green dot
- Tasks are ordered by priority, top to bottom
- Today is marked with an arrow; for the project to be on schedule, all tasks above the arrow should contain green dots

How SQA Fits In

- SQA components
  - Quality control performed by the Development Team
    - Focus on component design with test-first software
    - Unit tests are captured and automated
    - Unit tests are checked in with the code
    - An automated unit test framework such as JUnit is used
    - Benefits of quality control:
      - Immediate feedback on whether code violates design assumptions in the overall system
      - Testing is an act of design
      - Best and most effective means of documenting the design
  - Quality assurance performed by the QA Team
    - Decide if the software meets its intended purpose
    - Determine if the system in its intended configuration scales and handles the expected load
    - Provide insight into user interface design by creating edge use cases

QA Team

- Collocated with the Development Team
- Sometimes QA engineers are paired with Developer Pairs
- QA Team involvement in the process:
  - Reviews story cards containing business values (requirements) – are the business requirements measurable?
  - Listens to developers as they discuss task estimation
  - Stakeholder in the scheduling process, advocating the ordering of tasks
  - When the QA Team validates a green task, the team puts a gold star next to the task (it's ready for release)
  - By the time a group of iterations is ready for release, the QA tests have been run many times
  - Once software is released, the QA Team writes up story cards with errors (defects) that need to be fixed

Why the Menlo Software Factory™ Approach?

Why the Menlo Software Factory™ Approach?

- Eclectic software development approach
- Logical way to handle 2 software “evils”
  - Requirements
  - Scope creep
- No monopoly on expertise within overall Development Team
- Attempt to remove as many dependencies amongst code as is possible, diminishing the need for implementation order
- Use of sound anthropology and psychology principles for addressing both what is to be done and how it can be done
- Team decides on schedule; not dictated to them
- Project is defined by both what must be done and the estimation of how long it will take
- Prioritization of schedules left to the business stakeholders
- Customer satisfaction and user needs addressed through representation of users by skilled business analysts, and usability and UI design experts
- No need for lots of project artifacts; it’s possible to remove the need for customer training and sometimes most user product documentation

Education Crisis

Education Crisis

- 1920s & 1930s, 1950s & 1960s, 1970s & 1980s – reform movements because students are not measuring up to past generations, to students in other countries, or they are ill-prepared to take over from the previous generation
- After W.W. I - philosopher John Dewey said American education must be more child-centered
- Noted psychologist Jean Piaget demonstrated that it is the student who must be the principal agent in his own education and mental development
- 1948 – Columbia sociologist Robert Merton said that until teachers raise their expectations, students will continue to perform to low expectations
- 1957 – Sputnik launch caused American education to put an emphasis on covering more information, working harder, teaching more, and homogeneous grouping of students; education became subject-centered
- 1958 – noted anthropologist Margaret Mead suggested that we stop the vertical dissemination of knowledge and concentrate on lateral knowledge across the entire population
- 1966 – Coleman Report showed that class size, per pupil expenditure, number of text books, and other school inputs have less an effect on academic performance than a student's family and social environment

Education Crisis (con't)

- 1967 – Plowden Report in England showed that informal education has a much larger impact on the preparation of students than a formal education
- 1969 –William Glasser published Schools Without Failure – public schools are designed for the failure of students from elementary school through graduate school
- 1970 – Charles Silberman published Crisis in the Classroom: The Remaking of American Education – the American educational system is characterized by the failure or refusal to think seriously about the purpose of education and to question established practices
- 1970 – John Holt wrote How Children Fail – nearly all students fail in school because they are afraid of failure, bored, and confused by what they are taught
- 1989 – Carnegie Educational Institute published "Turning Points: Preparing American Youth for the 21st Century" - middle schools are failing to teach adolescents to prepare them for living in the 21st century

Promising Educational Reforms

Promising Educational Reforms

- John Dewey's questions in the early 1920s and reiterated by Charles Silberman 50 years later:
  - What is education for?
  - What kind of human beings and society do we want to produce?
  - What subject matter, classroom organization, instructional methods are needed to produce the desired results?
  - What knowledge has worth?
- After the subject-centered curricula development of the 1950s and 1960s by researchers and scholars unconnected to public education, Silberman points to a blend of child and subject-centered education
- Transmission of knowledge laterally to all
- Abolish homogenous grouping
- Abolish A-B-C-D-F grading systems, especially in elementary and middle schools
- Concentrate on carrying on the inroads made in pre-school programs for low economic and high stress home environments through middle primary grades
- Make upper level students responsible for their own education
- Students will rise to the level of expectations educators have for them

Promising Educational Reforms (con't)

- Revamp teacher education programs and teach teachers as they will teach students
- Specific techniques, methodologies and changes:
  - Classroom meetings
  - Learning centers
  - Activity cards
  - Individualized instruction
  - Computer-aided instruction
  - Open classrooms
  - Informal educational environment
  - Busy, noisy, stimulating learning environment
  - Flexible modular scheduling
  - Elimination of stringent school rules in upper grades
  - Abolition of harshness and unkindness
  - More open-ended discussions with no answers
  - Problem solving of "real problems"
  - Critical and creative thinking
  - Inquiry methodology replaces information dissemination

Parallels Between Software and Education Crises

Parallels Between Software and Education Crises

- Crisis identification with partial solution implementation occurs in cycles
- No silver bullets
- Problems have remained even after applying changes
- There is hope for composite solutions
- Common solutions between education reforms and the Menlo Innovations Software Factory™ methodology:
  - Know your stakeholders (society, company)
  - Know your users (relevant subject matter and products, simple user interface)
- The teacher/manager is a coach/facilitator
- The working/learning environment is open, collaborative, noisy, busy

Parallels Between Software and Education Crises (con't)

- Participants are accountable
  - Students are accountable for their own learning
  - Software developers are accountable to the schedules they have estimated
  - The Voice of the Stakeholder is responsible for managing the project with the stakeholders
  - The Voice of the Users is responsible for creating the user interface on the killer application that explicitly understands how the user does his job
- Students and software developers have standards to user for guidelines, structure, and collaboration
- Grouping of students and project teams is heterogeneous
- Evaluation is either by self or peers
- Process is meant to be a vehicle and not the governing element
- Frequent group status checks and easy movement from task to task
- Attention paid to the entire project, curriculum, student
- Functioning features are completed and presented at each iteration (system is fully integrated and installable)

Conclusions From My Personal Experiences

Conclusions From My Personal Experiences

- From education and training experiences:
  - Subject matter must be relevant
  - Teacher/instructor cannot focus on the dissemination of knowledge
  - Students/audience is responsible for their own learning
  - Total lecture is rarely the best method of learning
  - Learning styles do make a difference (not all people are auditory or visual learners)
  - Classroom discussions around open-ended questions are the most rewarding for all
  - Collaboration fosters decision-making and the ability to make decisions
  - Heterogeneous grouping is much more enriching than homogenous grouping
  - A rich environment in an open classroom with activity centers, small group activities is needed
  - Visual materials supplement experiences, activities, and discussion
  - Grades do not drive learning
  - True team-teaching is advantageous
  - Teacher preparation must include an understanding of the history, philosophy, and sociology of education, anthropology, psychology, and subject matter

Conclusions From My Personal Experiences (con't)

- From my own software development experiences
  - The software development process cannot drive the development but must be a vehicle for product development
  - The goal of development must be to create what the user needs when they need it
  - The Voice of the User must be represented by people who can truly understand how the users get their jobs done
  - Usability and user interface specialists who are experts in the areas of user interface design and human factors are the best ones to design the system interface
  - Stakeholders and users are not the same people and should be represented by different teams
  - A project team cannot afford to work as separate disjoint functional teams
  - Developers and QA engineers must come to an agreement on terminology so that the "features" delivered by software engineers match the "features" the QA team tests
  - Pairing developers and changing pairings frequently eliminates the monopoly in product areas that can hamstring software projects

Conclusions From My Personal Experiences (con't)

- Insisting on automated unit tests is necessary with frequent builds and iteration releases
- Quick daily project team status meetings, as employed in the SCRUM agile methodology, provide much value in getting issues and questions raised immediately
- Even though an open environment is busy and noisy, it is the best environment for quick resolution and "what if" scenarios; it is the best way for teams to easily communicate and cuts down on long involved email trails
- Open posting of status, key practices, and design work in progress is a good way for all to get a clear picture of progress, including stakeholders and users

References

References

Menlo Innovations. http://www.menloinnovations.com
Richard Sheridan. President, Menlo Innovations. rsheridan@menloinnovations.com

Richard Sheridan. “Secrets of Software Success: Adapting Projects to an Accelerated Society”, 2004. http://www.menloinnovations.com

Richard Sheridan. “Secrets of Software Success: The Nature of the Team”, 2004. http://www.menloinnovations.com

William Glasser. Schools Without Failure, 1969. Harper & Row Publishers, Inc. New York, NY

Charles Silberman. Crisis in the Classroom: The Remaking of American Education, 1970. Random House, Inc., New York, NY

References (con’t)

Carnegie Council on Adolescent Development, Carnegie Corporation of New York. “Turning Points: Preparing American Youth for the 21st Century”, 1989.

John Holt. How Children Fail, 1970. Dell Publishing Co, Inc. New York, NY.