Menlo Innovations

Secrets of Software Success:

The Nature of the Team

Richard Sheridan © 2004
President,
Menlo Innovations

Menlo Innovations
212 N. Fourth Ave
Ann Arbor, MI 48104
Phone (734) 665-1847
Fax (734) 665-2990
www.menloinnovations.com

The Nature of the Team

The typical delivery team has the following staffing profile: one project manager and three to seven software developers. Unfortunately, this profile only describes the *Development Team*. A profile that consists only of a *Development Team* sets the entire project up for failure at the very start. The Development Team is very important, but this team artificially elevates the voice of the developers to be the primary voice on the project.

A Development Team operating alone, directing projects without feedback from stakeholders and users, results in disaster. In the year 2000 over 67 billion dollars was wasted on failed IT projects. This paper teaches how to build a complete *Delivery Team* not just a *Development Team.* A *Delivery Team* working in a functioning Software Factory fills the gaps between business values, user values and development values. A Software Factory delivers software projects successfully, every time.

This is the second paper in a series on how to address the current crisis in IT. This series is both blunt and practical. Our goal is to provide a clear path to positively change your development team and your entire corporation. These papers introduce concepts vital to the future of our industry.

Our first paper, Secrets of Software Success: Adapting Projects to an Accelerated Society, introduced the metaphor of *The Software Factory* and why it is essential for survival in our modern environment.

This second paper, Secrets of Software Success: The Nature of the Team Team, builds on the first by describing the different teams in The Software Factory, their roles and responsibilities and how to assemble a true delivery team.

I'm glad you've chosen to download this paper. I believe you will find it both engaging and informative. And, if your experience is like mine, you just may find these ideas life changing.

Keys to Success

Before diving into our description of what a successful delivery team looks like, I want to ask a question. "What makes a software development initiative successful?" Or, perhaps more importantly "What causes so many software projects to fail?" These are reasonable questions to ask and the focus of an almost 10 year study by the Standish Group. The Standish Group has boiled down their research to the following list of factors that make software projects successful. The factors, from their year 2000 study, are listed in order of importance: [1]

1. Executive Management Support
2. Users Directly Involved
3. Experienced Project Managers
4. Clear Business Objectives
5. Small Milestones
6. Standard Software Infrastructure
7. Firm Basic Requirements
8. Formal Methodology
9. Reliable Estimates
10. Other

Software development processes must be designed to achieve these success factors. Everyone has a process; even if they have never thought about it or documented it they are following some sort of process. The question is this: is the process working to achieve the success factors or is it working against them? The Standish Group also adds a weighting to the success factors to capture the relative importance of the items; the larger the number, the more important the item to successful software development.

1. Executive Management Support (18)
2. Users Directly Involved (16)
3. Experienced Project Managers (14)
4. Clear Business Objectives (12)
5. Small Milestones (10)
6. Standard Software Infrastructure (8)
7. Firm Basic Requirements (6)
8. Formal Methodology (6)
9. Reliable Estimates (5)
10. Other (5)

[1] The 2001 Standish Group Chaos Report.

Look at the list. A great programmer, the focus of most organizations for success, doesn't even make the list - but Users Directly Involved is #2.

The most competent programmers in the world will fail miserably if they build software that the users don't need or can't use. Poor programming skills are not causing most failures; out-of-date development practices are causing most failures: not involving users, not keeping executive support and not having clear business objectives.

Software is created to achieve business goals. Yet most development practices have isolated the business and development teams. As a result, requirements are communicated without business needs in mind. Users are not represented at all. Schedules and milestones are grossly optimistic. All of this leaves the programmers to decide what is and is not important to the business. Is a programmer rewriting your marketing plan? How strong is the link between your business team and development team? Is there a link or a gap?

To understand why one software development process works better than another takes an understanding of business, psychology and engineering. If we were all perfectly logical, software development would be easier. A successful development process must understand how people actually work, learn, interact, grow, succeed, and fail.

Dr. W. Edwards Deming called this type of understanding Profound Knowledge. It includes the understating and application of psychology, including people's needs and fears; all of which is a tall order for a development process.

A Typical Team Profile

The Software Factory solves the problems of failed software projects by working with, not against, the nature of developing software in an accelerated society. The Software Factory solves the problems of failed software projects by making sure the process directly addresses the software success factors identified by The Standish Group Chaos Report.

However, it is not immediately intuitive to most people why The Software Factory works. In fact, many of the practices are initially counter-intuitive. In an industry, that has a rate of failure as high as our industry *conventional wisdom* usually isn't wisdom.
The typical delivery team we encounter has the following staffing profile:

- 1 Project Manager
- 3 to 7 Software Developers

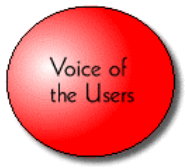
Unfortunately, this profile describes only the *Development Team*.

A profile that consists only of a *Development Team* sets the entire project up for failure at the very start. The Development team is very important. But having only this team artificially elevates the voice of the developers to be the primary voice on the project. The developer voice is the voice of technology. The developer voice presents a programmer's interests, concerns and world view. It is not sufficient to have this view directing the project. If this team is working without additional input the results it produces will likely have no business value. Even when the results are inventive, creative and revolutionary, they will most likely still be useless to the business.

Voice of the Stakeholders

Building a team entirely of developers omits important voices the delivery team requires. These voices are unrelated to technology and engineering. For example, who on your delivery team can properly represent the project's stakeholders? Certainly not the developers. Who on your delivery team sets the project's vision? Who on your delivery team understands the business objectives? Who on your delivery team knows the expected return for the dollars invested? The developers or the technologists? I hope not. The typical software project does not sufficiently represent the stakeholders' interests. They are voiceless. Theoretically, the projects are being built for the stakeholders. Practically, the stakeholders have little true ability to direct, impact or change the project once it is going. Do your stakeholders have any controls other than granting and removing budgets? Are your stakeholders intimidated by the cult of technology? Are your stakeholders silenced by the developers' convoluted descriptions of technology? Do your stakeholders have a voice? They absolutely require one.

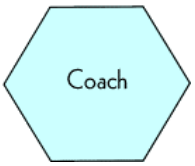
Voice of the Users



Building a team entirely of developers also omits input from the system's users. How are the users' interests represented on your projects? Is the system usable by real users? Remember that real users are far less comfortable with technology than the software developers who are designing and building the technology. Is your development team taking into account how users actually work in their real environment? Many systems we encounter are a great demonstration of technological prowess but they are unusable by real users. They don't meet real

users' real needs. We've literally heard developers joke about how stupid users are. The developers believe that if the users were a little smarter they would understand why the system the developers built is so great. Notwithstanding the fact that the system doesn't do what the users need it to do. How do many developers interpret the fact that users can't understand the software? The developers say, "Give the users more training". This can also be interpreted, "Teach the users how I think". Does the software you create intimidate your users? Can your users quickly be productive with your software? How do you know? Users need a voice on your delivery teams. Do they have one? They absolutely require one.

The Coach

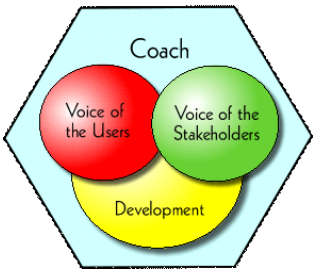


A working delivery team is a team that properly captures and represents the multiple interests on the project: stakeholders, sponsors, users and developers. Each voice must be balanced by the insights and concerns of the other voices. To provide that balance requires a mediator, a coach, for the team as a whole. The final component missing from most teams is someone who knows how to make the various voices work together: someone who unites multiple voices into a single functioning team.

The elements of a successful delivery team fill the gaps between business values, user values and development values. A properly constructed *delivery team* has multiple parts of which the development team is an important part, but only one part.

The Software Factory Team

The Software Factory consists of four teams working in concert to deliver projects successfully. These four teams are represented in this figure:



The focus areas have different and distinct responsibilities:

1. Voice of the Stakeholders - Representing internal business goals.

2. Voice of the Users - Representing user goals and abilities.
3. Development - Writing the software.
4. Coach - Making the voices work together.

These teams leverage different best practices from a variety of sources including:

1. The Unified Process
2. Interaction Design
3. Six Sigma and Lean Manufacturing
4. Extreme Programming

Each voice is a unique team. Each team requires specific specialized skills to be mastered by its' members. Each team requires dedicated, highly trained specialists. The Software Factory team members are not permitted to be more than one voice.

The environment in which the teams operate is also different. Core practices include:

- Working together in an open and collaborative production area.
- All four teams run simultaneously, throughout the entire project.
- The teams deliver a completed project every two weeks.
- The teams develop the software iteratively and incrementally.

The practices are in place to make the four different teams as effective as possible. Before studying the teams in isolation, let's briefly take a look at the teams in action with a Software Factory tour.

A Factory Tour

Upon entering a Software Factory it is immediately apparent to most visitors that things are very different.

Inside the Software Factory

Planning Meeting inside the Software Factory

Software Factories are large, open and accessible. There are no dividing walls and there are no cubes. The floor plan is flexible, broken into temporary work areas by long folding tables that are clustered in groups of three or four. On each table sits a single computer and in front of that station sits two developers. In the Menlo Software Factory there is only one computer for every two developers. All software development is done in pairs.

Today is the first day after a new iteration has begun. As you look around the room you notice a variety of different behaviors. At one station one person is typing and talking while another is reading the code being typed, occasionally making comments or suggestions. At a white board at one end of the room a group of four appears to be talking about software design, a class diagram appearing on the board. At a cork board at the other end of the room one person is describing screen shots and user behavior to two more people. At yet another workstation three people are seen sitting together, two looking on as one is typing at the keyboard. And at yet another station one person is typing while the other person is controlling the mouse.

Paired Programming inside the Software Factory

The Factory is busy, alive and energetic. All four teams, Voice of the User, Voice of the Stakeholders, Coach and Development are in attendance. There is also a fair amount of noise but work is clearly getting done. Extreme Programming as it is practiced in the Software Factory has a simple rule about paired programming – a pair must type all production code into the system.

Paired Programming inside the Software Factory

In the Software Factory the coach at the beginning of each iteration reassigns the pairs. The pair is expected to work together for that iteration to accomplish the stories that they have been assigned. The coach has assigned pairs to insure that no one on the team becomes the only person who knows how to do something. The coach is also assigning pairs to mix talents, experiences and skills and facilitate communication across the team. The assigned pair is free to break up and assist other pairs as needed, but they still are expected to complete the tasks they committed to deliver.

There are many benefits experienced by teams that do paired programming. And we are always quick to point them out to our visitors. These benefits include:

- All production code is always reviewed for bugs, clarity, simplicity and compliance with code programming conventions.
- All production code is designed and understood by at least two team members.
- All production code is made robust. At least two people have concentrated on trying to identify how the code may fail. They have designed, built and run an extensive set of unit tests for all production code.
- Team members always and automatically share skills, tips, patterns, idioms and experience.
- New team members can be easily integrated into the team.
- Fewer meetings are required.
- Fewer bugs are generated.
- The team is never held hostage to the ability or knowledge of any one individual.

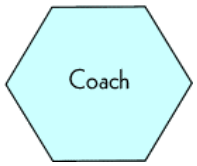
Located in the same room with the developers are the coach and high-tech anthropologists. Since it is the beginning of a new iteration, analysts and usability specialists are actively explaining to the developers the screens and behavior of the system that is to be built and completed in the next two weeks.

The Software Factory is transparent. At the far end of the room is The Planning Table where an Interactive Planning session was run the day before by the Stakeholders. It displays clearly on the wall in a Status Display what everyone is working on and the current status. It has one goal, the successful production and delivery of software that provides real business value.

Not readily visible when visiting the factory are the four different specialties that are at work at any given point in time. In the Software Factory, change begins with understanding why we need the multiple specialties. Hopefully it is already

becoming clear. Building a truly effective delivery team will require retooling your entire organization.

You absolutely require a Coach, Voice of the Stakeholders, and Voice of the Users team. You must learn how they operate and you must begin to staff and train your specialty teams.



The Coach

The Coach is responsible for the process as a whole. The coach ensures that all of the teams and the processes are properly organized throughout the life cycle of the project. Coaches are responsible for the productivity of the teams, the scaling of the teams, and layout of the development facilities.

In context of the Standish Group report, the coach is responsible for 30% of the success factors:

- Experienced Project Manager (14%)
- Small Project Milestones (10%)
- Formal Methodology (6%)

The coach may be a single methodologist or a coaching team consisting of a methodologist, quality assurance manager, and project manager.

In the Software Factory the Coach manages the deployment of resources between multiple projects working closely with the Voice of the Stakeholders to ensure that each project's goals are met. The coach is responsible for the following:

- Effective communication
- Stability in the midst of change
- The Facility
- Improving the process

A coach is always present in the Software Factory facilitating smooth factory operations.

Effective Communication

The coach establishes the communication paths required in all Software

Factory projects. The coach is constantly monitoring communication methods seeking new ways to improve it between the teams. The coach understands that written documentation can be inadequate and lead to misunderstandings. So additional formal communication paths in the Software Factory are established through a series of formal "ceremonies."

The events are called ceremonies instead of meetings. They are ceremonies because they run in a ceremonial fashion - a formal set of acts performed as prescribed by custom. The coach establishes the customs and the ceremonies are scheduled regularly. The Software Factory ceremonies include:

- Interactive Planning Sessions
- Daily Stand-up Meetings
- Show and Tell Presentations

Interactive Planning Sessions

The Coach runs the Interactive Planning ceremony. Interactive planning is scheduled the day before each iteration begins. During interactive planning tasks are written, re-estimated, prioritized, and assigned to developers. The Development Team does the estimation. The Stakeholder Voice does the prioritization, and the Coach assigns specific stories to specific developer teams. Each team has real and specific responsibilities.

Interactive Planning is the key point of contact between the Voice of the Stakeholder, Voice of the User and the Development Team. Implemented properly, Interactive Planning and Iterative and Incremental releases facilitate achieving 65% of the success factors:

- Actual releases manage expectations and keep executive support 18%.
- Actual releases create excellent opportunities for user involvement 16%.
- Provides feedback opportunities to clarify business objectives 12%.
- Provides small, measurable and verifiable milestones 10%.
- Provides opportunities to improve estimates and estimation abilities 5%.

We will look more at how interactive planning meetings run when we describe the Voice of the Stakeholders team.

Daily Stand-Up Meetings

This is a short and simple meeting. At precisely 10am each day in the software factory an alarm sounds. The ceremony is always called by an automatic alarm because people have a hard time interrupting others to start a meeting and the alarm doesn't mind being rude. The alarm, therefore, gets the meeting started on time. For the duration of the meeting everyone remains standing in a big circle. Remaining standing reminds people the meeting is supposed to be short. An item is select to be the speaking token. Only the person holding the token is allowed to speak. The token is passed only once around the circle and only the person holding the token is allowed to speak. When you receive the token you say:

- What you have accomplished today
- What you are working on
- Any questions you may have

Questions are answered one-on-one only after the stand-up meeting is over. The meeting is short and sweet. It keeps the team informed of progress and facilitates quick and controlled dissemination of team problems and status. Our meetings never run longer than 13 minutes for teams of up to 40 people.

Show and Tell Presentations

Show and Tell presentations are done at the end of each iteration. At a show and tell presentation the developers are asked to install their software onto a production system and demonstrate the working functionality live to an invited audience. Everyone is invited to attend. This presentation helps developers communicate to:

- Other developers
- The User Voice
- The Stakeholder Voice
- Executive, sponsors, stakeholders and users

The developers demonstrate their completed work, live and in person. The progress, or lack of progress, is clearly evident to everyone. In the Software Factory there are no secrets and no surprises.

The Facility

The coach is responsible for procuring a proper facility for a Software Factory. Central to establishing a Software Factory is an open area that allows developers, coaches, stakeholders and user voice members to work together easily. Team members interact without having to move from location to location, schedule meetings or delay responses. Team members simply roll their chairs from one table to another within a single room to facilitate interaction within the team. Those working in the capacity of user voice will often visit with the end users and domain experts to seek better answers to questions that developers raise during development.

The Coach also reinforces the practices by how the workspace is decorated. Typically, reminders of key practices, software metaphors and status displays cover the walls. The coach uses the environment to equip the team, helping it to learn and implement the core practices.

Stability in the Midst of Change

One of the sweet ironies of the Software Factory is that it embraces change by creating an environment of stability unprecedented in the practice of developing software. Stability and change are opposing forces. The business needs to be able to change rapidly as the environment changes. The developers want to be able to finish what they are working on and do quality work. The struggle is to build a development process that embraces change giving business what it needs while providing a stable and productive environment for engineering. The Software Factory accomplishes both. This is what is stable in the Software Factory:

- The development practices.
- An assigned task.
- The daily stand-up meeting.
- The interactive planning meeting every iteration.
- The Show and Tell meeting every iteration.

It is essential in the software factory that task assignments be stable. Developers are permitted to work on tasks without interruption for the length of an iteration. In The Software Factory two weeks is the longest iteration we allow. Smaller projects may have smaller iterations.

There are often a variety of random inputs from customers including new requirements, fixes and enhancements. In the past, these were always firefights

causing major disruption to any development schedule. By selecting two-week iterations we avoid these firefights by asking if the customer's issue could wait until the next iteration. On average, this would be a one-week wait. Most often, the customer can wait. The times they can't, we adjusted accordingly, but those instances are few and far between.

Staff and Study

The coach must be trained in all of the disciplines required in the software factory and must understand the activities of all of the other teams. They should, by temperament, be voracious readers.

From a process point of view we ask them to study Project Management, Extreme Programming and The Rational Unified Process. Coaches are typically the most knowledgeable people on the team. What is exciting about the software factory is that it allows an exceptional coach to be properly leveraged over multiple projects.

Our coaches are required to read all of the books recommended to all of the specialty teams. Books specific to the coaching staff include:

1. The Project Management Body of Knowledge from the Project Management Institute.
2. The Rational Unified Process by Grady Booch, Ivar Jacobson and James Rumbaugh.
3. Re-imagine by Tom Peters.
4. The Diffusion of Innovation by Everett M. Rogers.
5. First Break All The Rules by Marcus Buckingham and Curt Coffman.
6. The Innovators Dilemma: by Clayton M. Christensen.
7. Out of the Crisis by W. Edwards Deming.
8. The Six Sigma Revolution by George Eckes.

Voice of the Stakeholders Team

The "Voice of the Stakeholders" is responsible for delivering 36% of the success factors identified in Standish Group report. Specifically they are responsible for delivering:

- Executive Management Support.
- Clear Business Objectives.
- Firm Basic Requirements.

Of all of the teams in The Software Factory the "Voice of the Stakeholders" is unique because it is a virtual team. It has very specific responsibilities in special ceremonies facilitated by the Coach. These ceremonies occur regularly, but on any given day at the Software Factory the "Voice of the Stakeholders Team" may have no specific members present. During Stakeholder Ceremonies the following team members will be present:

- The Coach
- A "Voice of the Stakeholders" Team Lead
- Other Managers who may also be stakeholders themselves
    - Sales representative
    - Marketing representative
    - CEO, VP's, etc.
- A "Voice of the Users" representative, to be described later.

This team is under the direction of the *lead* who is empowered to make specific leadership decisions and direct the entire project. We refer to the "Voice of the Stakeholders" team as the Pilot House that is responsible for steering the development ship.

The Queen Mary or the Titanic

The engine room on a large ship is an amazing site. A working engine room feels like the living heart and soul of a ship. An effective development team can hum like a well-tuned engine room. We equate a team of highly skilled software developers to a good engine room. The big question is, are they the engine room of the Queen Mary or the Titanic? Think of icebergs as business reality. Icebergs come in all sorts and sizes:

- Wrong Product Vision
- Surprise Competitive Announcements
- Changing Management
- Political Infighting
- Mergers and Acquisitions

An effective developer team that can estimate, test, code, design and deploy software may still fail. The perfect engine room still goes down with the Titanic. It isn't fair, but it is true. The Titanic's engine room sits at the bottom of the Atlantic not because of problems in the engine room but with problems at the pilot house.

The Pilot House

The "Stakeholder Voice" Team is the Pilot House. This team provides project ownership. It is their project and the project succeeds or fails based only on the quality of their decisions. They are the project owners.

The Pilot House identifies and prioritizes the features to maximize the return for the business. Their goal is to build a system that is profitable for the company. Specific skills mastered by members of this team include:

- **Stakeholder Identification**: Identification of the project stakeholders within the business.

- **Business Case Creation:** Identifying the business reasons and ROI expected if undertaking the software development initiative. This provides the clear business objective identified by the Standish Group as essential for successful projects. If clear business objectives cannot be found the project is cancelled and considered a success. It is a success because it was cancelled efficiently.

- **Task Prioritization:** Identify which tasks should be done and in what order. Specifically, what tasks are most important to be done next to meet business goals and objectives? This is important to maintaining executive management support.

Demonstrable Progress

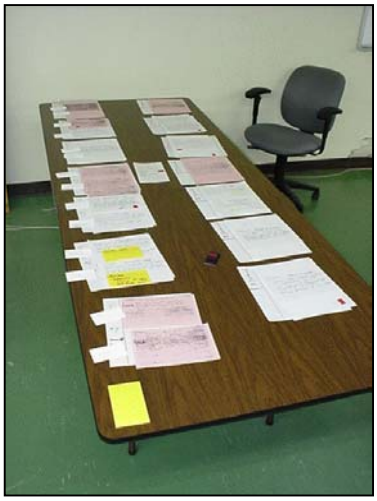
In order to steer the ship properly, the Stakeholder Voice must have a closed loop relationship with the development team and the stakeholders. A closed loop with developers means that the Stakeholder Voice clearly understands development estimates and can frequently see demonstrable development progress. The only

progress accepted as demonstrable is a working program.

A closed loop with the actual stakeholders means that the Stakeholder Voice can clearly describe the resources, scope and time of the engagement to the stakeholders. This is presented as a project plan.

The Project Plan

The project plan in The Software Factory is designed to be highly entrant and participatory for the team members. The team can quickly play *what if* games to understand how resources scope and time interact to produce the desired results. The project plan itself consists of rows of sheets of paper laid out on a table and covered with tasks.



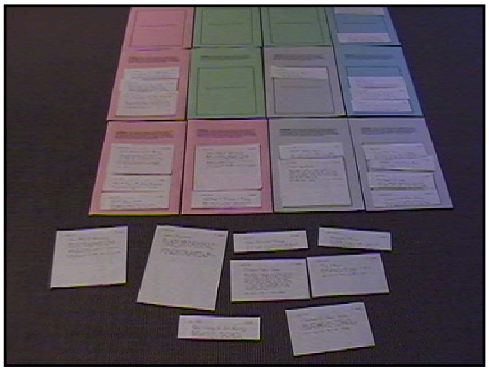
The Planning Table

- A row represents the development team size working on any given iteration.
- The multiple rows represent the number of iterations in the project plan.
- The folded task cards represent work to be accomplished.

After the developers have estimated the work, the Coach creates a consensus estimate and the estimate is written on a task card. The card is folded into a specific size proportional to the estimate.

Planning the Work

In the Interactive Planning session the folded sheets of paper representing tasks are placed on larger sheets of paper representing team capacity. When the large pieces of paper are full, the plan is complete.



Of course, not everything always fits on the plan.

This is intentional. In software development true planning involves making trade-offs, this especially includes what not to include in the plan.

The Status Board

The team maintains a status board in the Software Factory to demonstrate work progress to all delivery team members – but especially to the stakeholders. All team members can easily track and understand progress of every project.

The status board is a duplicate of the tasks in the project plan for the current iteration. Unlike the project plan, the status board also identifies the specific team members responsible for delivering a task during the iteration.

Stakeholders know that status is never a secret. Clearly displayed are:

- Tasks assigned for the iteration.
- Who is assigned the task.
- What tasks have been started.
- What tasks have been completed.

The Status Board

The process is simple. When a task is started the team marks it with a yellow dot. When a tasks is completed the team marks it with a green dot. Tasks are assigned and done in order of importance from top to bottom. If the team is on track, all tasks above the line indicating today are marked with a green dot. It is simple for anyone, 24/7, to determine the status of the entire iteration. If tasks are blocked or behind schedule the status is clear to everyone. Then the Coach or The Stakeholder Voice team may then take the appropriate actions.

Releasing Software

The Stakeholder Voice Team releases software. They keep the true stakeholders on board with the project by continually showing real demonstrable progress.

The Stakeholder Voice is never blindsided to the development team because the development team, from the very beginning, is charted to produce releasable code at every iteration.

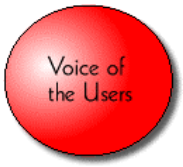
At the end of each iteration it is obvious to everyone which tasks have been completed because completed tasks are represented by working software. The work may or may not have immediate business value to the stakeholders, but if there is enough business value it is ready to be released: to the end users.

The development team only produces one type of measurable and verifiable deliverable, releasable code.

The Stakeholder Team decides when enough functionality is present in these deliverables to be of value to the business and the users.

At that point in time the software is released.

The Voice of the Users Team



The Software Factory's "Voice of the Users" team utilizes a practice we call High-Tech Anthropology™. High-Tech Anthropologists specialize in understanding corporate cultures. They use their knowledge to improve the process of designing, creating, introducing and adopting new technology.

High-Tech Anthropologists study the cultural context of systems to understand real user needs and express these needs in new system design. By obtaining a better understanding of the users and their environment, the high-tech anthropologists are far more successful at eliciting requirements and designing usable systems. As a result, new systems are significantly more successful because the user community can effectively understand and utilize them.

The Voice of the Users team captures requirements clearly and reflects those requirements in user interface design and detailed instructions to the development team. As High-Tech Anthropologists, they have also specialized in understanding engineering cultures and how to capture and communicate requirements effectively to the development team. The Voice of the Users team works closely with real end users and engineers to ensure that user concerns are represented in technical decisions.

In context of the Standish Group report, the team is responsible for 22% of the success factors:

- Users Directly Involved (16%)
- Firm Basic Requirements (6%)

The High-Tech Anthropology teams play multiple roles. Let's review these roles one at a time:

Business Process Analysts
Business Process Analysts ensure that the business concerns are accurately reflected in the system requirements, by assessing the business goals of the organization where the system will be deployed. This includes:

- Understanding the personal goals and strategies of the project's sponsors.
- Understanding the personal goals and strategies of the project's users.
- Building models of the organization.
- Directing a business reengineering effort, if needed.

- Calculating the cost/benefit analysis for changes in the process.

User-Interface and Usability Specialists
User Interface and Usability Specialists create simple and practical user interface designs by understanding human factors and cognitive psychology. They are responsible for making the system easy and intuitive for the target users to achieve their business goals. They design highly usable user interfaces by conducting interviews and job shadowing of representative users. In the process they:

- Identify different types of users.
- Identify goals for each user type.
- Create personas for each user type.
- Identify the primary personas.
- Evaluate the system interaction design against the needs of primary personas.
- Conduct an evaluation against the basic usability principles.
- Identify the basic generic usability flaws.
- Create design mock-ups for revised user interfaces and storyboards.
- Conduct iterative review sessions of design mock-ups and storyboards with representative users.
- Collaborate with the development team as a "Voice of the User" which represents and interprets user needs to the development team.

The Mirror Persona

One area worthy of special mention is the creation and use of Personas. Personas are central to how our High-Tech anthropologists think about the user community. A persona is a specific, detailed description of a user type designed to make that user real and tangible. The user is given a fictitious name and is described in detail. The description is highly specific. The goal is to actually identify someone to design for.

Imagine you are building a web application for ten thousand users at a medical center. You might think that out of ten thousand users you couldn't possibly pick just a few personas to design for. Here is an interesting question. Is it possible to design interface and interaction for ten thousand people? Can you possibly keep the concerns of all ten thousand people in your mind at one time? The answer is, of course, no.

So, if you can't design for ten thousand people and you don't create at least one persona, then for whom are you designing?

The answer is simple, without personas, you will design for yourself, the Mirror Persona. So named for the person you see in the mirror each morning. Unfortunately for most projects, this means that the software is being designed for the programmers building the software. The very last people who will ever be using the software.

The team must design for a few personas that actually capture the real user community. If they don't specifically create them then the team will automatically design for themselves. We emphasize this point because it is the most common failure we observe on all projects.

Having ten thousand users demands that one focus on a few personas. The question is "will those personas be well defined or not?" Will the personas represent real archetypes of the user community or will the personas simply represent the development team?

Staff and Study

We intentionally do not staff The User Voice team with programmers. Not surprisingly, technologists do not relate to technology in the same way that users do. We seek to staff The User Voice team with people more akin to the real users. We train these people in analysis, requirement elicitation, usability principles and interface design. We do not train them in programming or writing code.

Good places to start looking for The User Voice team members are in your current customer support, technical writing and end user training departments. People in these departments are typically in the best position to appreciate how bad your current system really is for most users.

The following three books are required reading for those wishing to participate on The Voice of the Users team. We encourage The User Voice team members to study them in detail.

1. The Design of Everyday Things by Donald A. Norman. A great text on how most of the products around us are designed. If products are difficult to use, the designer has failed us. Learn how to design a door or a light switch before taking on something as complex as a graphical user interface.

2. The Inmates Are Running The Asylum: Why High Tech Products Drive Us Crazy And How To Restore The Sanity by Alan Cooper, Paul Saffo. Cooper's introduction to personas and our justification for not putting developers on the "User Voice" team.

3. About Face also by Alan Cooper. A great introductory text on user interfaces. This book gets down to practical examples and case studies.

Next Steps

Building a complete delivery team, not just a development team, is key to being successful for your entire organization. The change required to build a truly functional team is significant. The time to begin is today.

I'm Richard Sheridan from Menlo Innovations, hoping that you will have the capability and courage to change your teams and that you will learn to survive and thrive in the 21st century.

Good luck.

Menlo Innovations Training

Menlo provides multiple training classes for teams seeking to improve their performance. The classes listed below are specifically recommended to those interested in building and operating a Menlo Software Factory. Please give me a call at (734) 665-1847 and we can discuss your needs in more detail. Ask for Rich Sheridan.

Secrets of Software Success – 1 day

> The elements of successful software initiatives go beyond "best practices and tools." Business and technology are inseparable. Customers and business are inseparable. A successful software initiative must fuse customers, business and technology into a single functioning team: this course teaches how to build that team.

Software Project Management for Agile Teams – 2 days

> This class teaches how to manage an agile development team by simulating all of the activities found in a typical development iteration. It also teaches how to change how the development team interfaces to the rest of the organization in order to fully achieve the full benefits of agile development; flexible software, responsive teams and timely software releases.

High-Tech Anthropology 101 – 2 days

> This course teaches how to effectively capture requirements using the concept of High-Tech Anthropology™. It will teach you how to design software that your users will love. Participants practice writing use cases and performing other activities required for building high-value, user-friendly applications. Additional topics include: interviewing, job shadowing, persona writing and creating storyboards.

Menlo Innovations LLC
212 North Fourth Avenue
Ann Arbor, MI 48104
Phone: (734) 665-1847
Fax: (734) 665-2990
www.menloinnovations.com / rsheridan@menloinnovations.com