Ellen Braun
Guest Columnist

Lean/Agile Methods for Web Site Development

Lean/agile methods meld best practices from manufacturing—with team empowerment that would make an organizational development expert proud—and prototyping acceleration.

Methodologies, although lacking in glamour, are often the process backbone of significant Web development projects. Software engineering methods are currently evolving in ways that impact major Web sites, both developmentally and through ongoing enhancement. Lean/agile methods meld best practices from manufacturing—with team empowerment that would make an organizational development expert proud—and prototyping acceleration. These methods avoid some pitfalls of prior approaches to software development and implementation.

To fully understand the benefits of lean/agile Web site development, it is useful to review the maturation process of other development processes.

AD-HOC DEVELOPMENT

Many Web sites have their roots in an organic development process that was logical and useful for relatively simple sites. Well intentioned and technically literate Web site managers and teams built credible and effective sites based on common sense. They listened to stakeholders and employed a natural cycle for building and improving sites. Ad-hoc development, however, is not a method; it reflects the lack of a formal method. Advantages of ad-hoc development come from translating direct dialogue with customers into immediate action. When in-depth planning falls short during the project, quick rework patches things up.

Ad-hoc development works well for small and simple Web projects that have minimal interdependencies to other processes or systems. Sites of a few hundred pages, all generated from HTML, do not need the formality of a methodology to get a successful result. However, this changes when the sites become more complex.

Disadvantages of an ad-hoc approach are significant and painful. Fearsomely high failure rates of projects that involve implementation of new technology or other significant complexity result from:

- lack of clarity in sponsorship.
- unpredictable schedules and costs.
- morphing of requirements and drifting of overall project direction.
- high potential for "surprise factor" upon implementation caused by significant risks overlooked during development.

Projects with an infrastructure that makes sense from the perspective of each individual project often end up being suboptimal, expensive, or even impossible to maintain at an organizational level. The risks of applying no formal methodology to projects can multiply, driving consideration of formal methods for delivering Web projects.

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

SDLC and similar methods depend on a rigorous, planned flow of activities guided by analysis and thought. The approach can be applied to Web site development, system integration, and other significant software projects. SDLC assumes that the effective transformation of an idea to a new capability goes through predictable phases and that applying common processes at each phase can increase the success rate. The phases may sound overly regimented, but follow a natural evolution of ideas into execution: ideation, feasibility, definition, design, development, implementation, and close.

Business and technology personnel develop an idea and then carry out analyses to decide whether it's feasible. If not feasible, or if the economics don't make sense, the project is shut down. If feasible, the next phase aims to define the intent of the project by eliciting, understanding, and documenting what the particular capability is intended to accomplish in the form of high-level and detailed requirements. The design phase evaluates different approaches that enable these requirements to be satisfied. The construction or development phase includes working to build out the solution or to customize the implementation of a software package. This work occurs in alignment with the design. It includes testing to ensure that requirements are satisfied. Rollout or implementation puts the solution in place. The close phase wraps up the work and the product goes into maintenance mode. Each phase, until implementation, includes a step to refine the schedule and resource estimates, so timelines and costs can be understood with increasing precision over the course of the project.

Applied in its original form, each step creates a finalized work product that feeds into the next step, gradually leading to the completion of the project in a linear or waterfall manner. The approach proves to be methodical and reliable in many situations. Since the method is based on formal planning steps, execution is generally effective. Downsides include long project durations, heavy dependency of the creation of documentation as a primary communications method between team members, and lack of flexibility.

In many environments, SDLC and methods like it prove to be rigid, lacking immediate linkage to value. The linear basis of SDLC projects means that lessons learned along the way don't substantially revise the direction of the project. The approach assumes the original requirements were correct in the first place. My own experience is that the approach fits fewer projects than what is applied to it. Many senior project managers have had the unfortunate experience of delivering exactly what was specified only to find that customers weren't able to understand their own needs in a vacuum. The new capabilities are outdated by the time of delivery.

SDLC avoids some of the pitfalls of ad-hoc development by keeping tight controls on scope, careful project management, and applying the discipline of risk management to each phase of a project. Web projects that introduce a significant change in technology are good candidates for SDLC approaches, such as implementation of a content management system. Once teams learn the discipline of SDLC and deliver on commitments with reliability, they often start the search for faster approaches.

RAPID PROTOTYPING

Rapid prototyping is an iterative approach. As such, it acknowledges that developers and customers don't know what they don't know. Simply stated, the rapid prototyping process aims to quickly create a working model and then allow it to evolve, eventually to a production-ready state. The value of a prototyping approach is that it makes the project goals tangible. Direct conversations with the customer can discern requirements with detail, speed, and accuracy.

When it succeeds, the approach mixes the discipline of planning with the speed that comes from narrowly focusing on high-value functionality. The classic elements of effective planning, such as clarity of sponsorship, detailed planning and forecasting, accurate reporting of status, and rapid identification and resolution of issues, are required as a complement to the advantages of prototyping. Many Web site developers have success applying rapid prototyping to projects of small to moderate scale. Prototyping is also used to prove out a concept or fully explore an idea prior to significant investment in new capabilities.

The value of a prototyping approach is that it makes the project goals tangible.

Similar methods such as Rapid Application Development (RAD) and Joint Application Development (JAD) have elements of prototyping. These methods seek to accelerate project delivery by concentrating on the 80/20 rule and use effective group facilitation techniques to harness teamwork. Commercial Web development companies often apply a streamlined version of SDLC with RAD or prototyping to harness advantages of both approaches. Teams that apply these methods and find them useful but narrowly applicable have turned to lean/agile to gain a step change in ability to deliver.

LEAN/AGILE METHODS

"Lean" is a mind-set pioneered in Japan as an outgrowth of, and evolutionary step beyond, the quality movement. Lean applies at the level of an entire organization, the level of an individual process, and everything in between. The approach seeks to identify and eliminate sources of waste. Outcomes of applying lean approaches include faster time to market, higher quality, greater flexibility, lower costs, and empowered, effective team members.

In a manufacturing environment, the "seven wastes" initially identified by lean pioneer Taiichi Ohno of Toyota include overproduction,

➤

transportation, unnecessary inventory, inappropriate processing, waiting, excess motion, and defects. To apply lean is to continuously strive to eliminate these wastes in the value stream, as measured by outcomes important to the customer. Ideally, this transformation of the value stream stretches from raw materials to end customer.

Applied to Web site development and maintenance, the approach focuses on eliminating these types of waste:

- Overproduction: adding extra features
- Transportation: excessive handoffs, as with those between people with narrowly specialized skills
- Inventory: documenting and tracking the most detailed system requirements regardless of whether the functionality will make it into the final product
- Extra processing steps: addition of steps that don't add substantial overall value, such as documentation as a primary form of communication between individuals
- Waiting: for decisions as information is passed up the chain of command
- Motion: driving between locations for meetings
- Defects: fixing bugs not found in testing

Some lean advocates also describe the underutilization of talented resources, excess complexity, and task switching as key wastes in software development.

Lean thinking, applied to Web development projects, has features in common with its manufacturing roots. It emphasizes applying labor only to the most valuable functionality, empowering the team to make immediate decisions with streamlined customer representation, dividing the work into small chunks and focusing on them tenaciously, using testing and customer involvement to get immediate feedback on work products and eliminating waste. It results in some of the acceleration associated with rapid cycle development.

Agile is a specific application of lean concepts tailored to software development. Scrum is one type of agile development that applies well to Web sites. Although the word comes from rugby, the teaming approach is anything but chaotic and rough. Teams are relatively small (five–15 members), located together, and have well-defined roles (including scrum leader and product owner roles). The core idea behind this flavor of agile development is to set up a team that works together in a focused and fully empowered mode on the highest-value elements of the end product. If properly set up, the team acts as a self-organizing system, using a terribly simple interaction model that works only where it links directly with value to the business customer.

For example, high-level system capabilities are captured and prioritized in a list called the Product Backlog. The business decision maker is a full-time member of the team and has full accountability for the list, thus defining the emerging product. The team works in chunks 3 to 8 weeks in duration, called sprints. With facilitation from the scrum leader, the team comes to consensus to peel a certain number of items from the top of the list for the next sprint. This short list is called the Sprint Backlog. During the sprint, items on the Sprint Backlog are the only focus of work. Where possible, the team strives to build working models and test immediately, providing the rapid feedback value of a prototype. Learnings from one sprint often drive changes to the Product Backlog's content and prioritization. Reporting within a sprint is based on periodic assessment of progress against the Sprint Backlog.

To understand how agile methods deliver faster time to market, look at the things that *don't* happen within a project. Extra features are not added, since those with moderate or low priority are deprioritized in favor of valuable features. Time is not wasted on decision-making since the empowered decision makers are in the room together. All work happens within a sprint, and each sprint focuses only on a narrow set of tasks with greatest alignment to value for the customer rather than lots of concurrent efforts with unknown value. This avoids task switching.

The approach has interesting and high-impact implications for teams. To be effective from a skills point of view, team members must act proactively and often extend their skills. The method works best with people who are jacks-of-all-trades and masters of a few and with organizations that are not to rigid and hierarchical. From a cultural point of view, the method may not be compatible with all organizations.

Most successful and experienced Web site developers and managers have evolved to an entirely workable and useful fusion of many of these methods, often by taking some hard knocks along the way. Taking immediate action in response to a customer need is a key lesson from ad-hoc development. The discipline of accurate status reporting to identify and respond to issues is part of the legacy of classic project management put in widespread use with SDLC methods, helping to raise the success rate of bigger Web projects. So is the idea of actively killing off projects in early stages based on formal feasibility analysis. Using a prototype as a tangible focal point to elicit customer needs and iterating to incorporate the value of immediate feedback into the final Web site applies lessons from prototyping and rapid cycle development. Modifications of SDLC have incorporated prototyping and iterations in a variety of forms in order to address key weaknesses of classic SDLC approaches, often applied by commercial Web site developers.

Lean/agile methods provide additional tools. Concepts of key wastes and value stream can act like lenses to help each of us build and run teams which can consistently experience a great sense of flow that comes from delivering high-value Web capabilities.

Ellen Braun *manages a departmental IT team for a Fortune 500 financial services company. Her group applies lean / agile methods to large investment projects and to ongoing enhancements for a commercial software package.*

Comments? E-mail letters to the editor to marydee@xmission.com.