webXstream - an optimal methodology for web development

Tomasz Ceglarek, Tribal DDB Sydney, Australia TCeglarek@ddb.com.au

Xiaoying Kong, Faculty of Engineering, University of Technology, Sydney, Australia [HREF1] xiaoying@eng.uts.edu.au

Abstract

This paper presents an optimised web development process to assist web professionals in bringing a systematic approach to their efforts in developing web-based software systems. Firstly seeking a broader understanding of the needs of web systems, the characteristics of this medium were examined and assessment was made of current practices amongst web developers. Existing development processes were also examined to determine how they may apply to the issues at hand. An optimized methodology "WebXstream" was forged, utilising the optimal elements of several existing processes with extensions to facilitate the specific needs of the World Wide Web. This approach, based on the increasingly well-respected Extreme Programming methodology theoretically promotes efficiency through minimal administrative overheads, specifically caters for the inherent volatility of requirements in web projects by staging development over time and brings discipline to the practice without stifling the creative spirit that typically pervades such systems.

Introduction

The rate of failure with respect to web development projects is alarming. Studies suggest that as many as seventy per cent of these efforts fail to deliver on the agreed functionality (Johnson et al. 2001).

There is confusion as to the source of these problems, let alone potential solutions.

Empirical evidence does however suggest that systematic approaches are not being taken seriously, which could certainly undermine any engineering effort. With the potential for a 'web catastrophe' to rival the 'software crisis' in terms of client confidence, there is a need to reign in the "cowboy coders."

It is proposed that a dedicated web development methodology be created to cater specifically for the needs of clients, developers and users. It is to harness the potential of the World Wide Web in a manner that is most effective, efficient and appropriate.

This paper begins the task by first analysing the nature of the web as a medium, to appreciate what separates web projects from other engineering feats. It also examines what the causes of failure in web projects may be, and how methodology can assist with this.

The current spectrum of methodologies for software in general as well as the web are then examined in light of this scenario, to determine what aspects of these may be used to form an optimal web development methodology. Finally these aspects are synthesised and discussion of the implications of the optimal web development process is made.

Background and Literature Review

Characteristics of the web

There is a growing body of research that is attempting to understand the differences between the differences between web-based systems and conventional software systems. The unique characteristics of web-based systems can be examined in technical and organisational perspectives (Lowe et al 2001, Lowe et al. 2002), or art and science perspectives (Murugesan 2000).

At a technical level, web systems typically: have a tighter linkage between the business model and the system and information architectures; have more open and modularized architectures; use

technologies that change very rapidly; demand effective information design and content management; place more emphasis on user interface design; are more susceptible to security threats; and place increased importance on quality attributes in mission critical applications that are directly accessed by external users. At an organizational level, web systems typically: face high levels of client uncertainty of the system requirements and in understanding whether a particular design will satisfy their needs; have high levels of change in requirements and project scope, largely due to the mutual evolution of the business model and the web systems; have shorter delivery timeframes; demand finer grained evolution and maintenance with an ongoing process of content updating, editorial changes and interface tuning (Lowe et al. 2001).

Considering the perspectives of arts and science, web systems are largely document-oriented; The web product must include provisions for its own content, the science extends beyond the code to include development of content presented on the web - and the associated maintainability of this process; The look and feel of these pages are of utmost importance, being measured on a scale of usability (Murugesan 2000); The web information may be accessed by anyone through any connection theoretically at any time of the day; There is no defined entry or exit points from the Internet; a user may start at any page and end up at their desired destination by any one of an infinite combination of links, searches or other browsing techniques; There is no defined shape or organisation to the web or the information it contains (December 2003); There is tremendous competition for web site recognition by users. Sites must seek differentiation in content, design, structure in order to best satisfy the user experience (Nielsen et al. 2000); The web supports both a high degree of selectivity through user choices in hypertext as well as other possibilities for interactivity, such as user-to-user communication or customized responses to users through automated feedback mechanisms (Lynch et al. 2002); Web products must then cater for users of varied skill and capability (Murugesan 2000); Current web development efforts frequently bring together professionals from many varied backgrounds. These professionals have different expectations, experience and thus their contributions must be interpreted in appropriate ways (Powell et al. 1998, Burdman 1999).

These unique characteristics of web-based systems demand the change of development methodologies (Lowe, 2003).

Web ideals and aspects of an optimal web development methodology

The original brand of Internet hype painted this medium as an "information superhighway", a means of distributing information with speed and ease which would form the cornerstone of business in the 21st Century (Cyhaniuk 2003). As popularity of the service grew amongst the public, aesthetics took on a far greater importance. As a result, designers began to produce sites with a more elaborate artistic input, however this served to only frustrate users even further. Web development at present is attempting to find a mid point between these two extremes. Under the umbrella of 'web usability' there is considerable effort now being put into enhancing a user's on-line experience and this extends beyond simple design issues.

The optimal web experience has a number of aspects, for both client and user (Ceglarek 2003).

- The solution meets its functional and performance requirements, in line with client and user expectations.
- That the solution is efficient, that the requirements are met in the most useable and appropriate way.
- That the solution can be delivered in a timely fashion, and at reasonable cost.

The optimal web development methodology will require a balanced approach. On one hand, the rigors of traditional project management are necessary in order to lay a firm foundation for development, but the design process must be flexible enough to adequately capture the client's requirements, to thoroughly map these to an architecture or model without stifling the inherent creative nature of web design, and to finally be able to translate this model into a usable and reliable web product. Table 1 summarizes the aspects of an optimal web development methodology (Ceglarek, 2003).

Table 1: Aspects of an optimal web development methodology

Category	Aspect	Comments
Product	Usable	The site design must be learnable, efficient, memorable, cater for mistakes, be accessible to all and provide a feeling of satisfaction
	Timely	The content and structure must be up-to-date with user and client expectations, and be available in an appropriate time frame
	Sound technology	While utilising the latest technology may be attractive, the best design will use the most appropriate technology to the task
Management	Requirements aware	Allow for modelling of requirements of the web solution and its context, in terms meaningful to all parties
	Flexible	The project structure must be able to accommodate the frequent changes to a web system's problem domain as it evolves
	Documented	The design rationale and procedures must be sufficiently documented to facilitate effective development and maintenance
Design	Allow innovation	The artistry and creativity that pervades the World Wide Web cannot be stifled through inappropriate project regimes
	Economical	The product must be delivered at a reasonable development cost
	Professional mix	A multi-disciplinary approach may well be required to sufficiently cater for the wider needs of a web project. Professionals of all relevant disciplines must be included in the process, and it must cater for the ways in which they individually work

Literature review

Current methodologies take one of two forms: the classical 'engineering' or 'heavyweight methodology' and the newer 'agile' or 'lightweight methodology'.

Heavyweight methodologies employ traditional engineering management processes to software development, characterised by well-defined and largely linear planning, analysis, design, construction and testing phases, all bound by comprehensive documentation.

There are several drawbacks to this linear approach: problems are not identified until the testing stages, when it may be too difficult to costly to rectify; requirements must be fixed prior to any development and the design and code stage usually turns up scores of deficiencies and oversights (Barkstrom 2003)

Asserting that traditional engineering methodologies were inappropriate for the needs of software - being too restrictive, inflexible and carrying too much overhead in terms of planning and documentation - agile development methodologies such as Extreme Programming, Crystal Methodologies, Adaptive Software Development, Scrum, Feature-Driven Development were proposed.

Extreme Programming (Beck 2000) is centred on four core values: Communication, Feedback, Simplicity and Courage. These values pervade the entire development process, and manifest themselves through twelve practices(Maurer et al. 2002). Cockburn's Crystal Methodologies (ADM 2003) are notable for their people-centeredness, and by extension the latitude given to developers in finding what works best for them in a particular situation. Cockburn appears almost at pains to not inflict any sort of direct process or procedure, but merely provide 'properties' that the project should possess - leaving the team to decide how best to achieve them (Cockburn 2003; Jupin et al.2003). Jim Highsmith's Adaptive Software Development (Highsmith 2002) emphasises a new software lifecycle typically comprising three phases, speculation, collaboration and learning. Throughout, the emphasis is on embracing change as the need arises rather than attempting to control and eradicate it in order to maintain a predefined plan. In an adaptive environment "deviations guide us towards the correct solution." (Fowler 2003) Scrum is another agile

methodology that emphasises the need for comprehensive communication among team members and frequent delivery of working artefacts. There are a set of attributes and four stages to Scrum application. Focus on a manageable task is maintained, team communication is enhanced, obstacles are quickly identified and treated, and the possibility for conflict is minimised with centralised decision-making. Feature-Driven Development (Palmer et al. 2002) is a five-step process, feature-based development method for developing business critical systems focusing on the design and implementation phases (Abrahamsson et al. 2003). Rational Unified Process (RUP) is an infrastructure to assist in the implementation of a development process. The RUP can neither be classified as an agile or heavyweight process, owing largely to the flexibility in its design. In its purest form, the 'phases' it describes look much like those of a standard iterative waterfall process; this is heightened by the assertion that at the elaboration phase "While the process must always accommodate changes, the elaboration phase activities ensure that the architecture, requirements and plans are stable enough, and the risks are sufficiently mitigated, so you can predictably determine the cost and schedule for the completion of the development." (Gormik 2001) Macromedia's Fusebox Lifecycle Process (FLiP) gives a defined process to organise any particular web project around. There are eight steps to the FLiP cycle: Personas and goals; Wireframes; Prototype or Front-end development; Architecture; Fusecoding; Unit testing; Application Integration; Deployment. FLiP takes some elements of agile process. The structure is however still largely linear rather than iterative, and takes an alternative approach to handling change by almost eliminating the possibility of it after prototyping is complete (Ceglarek 2003).

These lightweight, or agile methodologies emphasise adaptation of product and process, embracing change as the project requires it - with as little procedural overhead as practical. With a number of variations, there is significant commonality between them on the following grounds (Laurance et al. 2002):

- Short development iterations, with products demonstrated to customers at regular intervals
- Continuous testing, as often as daily
- Collecting customer requirements using scenarios, stories or use cases
- Small team sizes, no more than thirty and frequently less
- Informal modelling techniques - CRC cards, simple drawings on whiteboards
- Adaptive planning techniques; plans are updated based on intermediate results and reprioritization by the customers

However the agile methods on their own are not sufficient. The unique characteristics of the web are not specifically accommodated in any one of these methodologies; not least because they are designed to generically fit any software development effort. Fusebox and FLiP go some way towards demonstrating how web needs can be built into a development methodology, however these solutions in themselves remain particularly platform specific. This is a disadvantage with a reliance on software tools to aid development efforts; tools are unlikely to cover all needs of every potential project. The ideal outcome would most likely lie somewhere between agile methods and fusebox-type methodologies. The inherent change appreciation, people-centred development, lightweight documentation and frequent structured communication, combined with practices to cover specific web structures and promote understanding of the problem domain for all involved would go some way to defining a usable and efficient process for web development.

Optimising methodology

Having considered existing methodology, its shortcomings and benefits, and the needs of a successful web system project these concepts they embody are now combined to form a framework for an optimal web development methodology.

Elements

Phases in web development

By comparing the phases of methodologies considered, the following phases emerge as common to most, and hence shall take place in the optimised model:

- Context analysis;

- Requirements modelling;
- Design and construction

Supporting processes

Along with the primary development effort, the following exists to support the work being done

- Estimation
- Documentation
- Communication channels

Model

The most suitable model for the optimal web development process requires both discipline and creative latitude for the developers. Extreme Programming (XP) is widely recognised for having the best balance between these two extremes (Beck 2000; Highsmith 2002; Maurer et al. 2002; Fowler 2003) at the cost of requiring a stricter regime to implement properly.

Given the attitude of "cowboy coders" it is intuitive that a more disciplined approach will offer a clearer path to the benefits of systematic development. Extreme Programming does not however burden the development under unnecessary overheads in terms of planning and documentation, and so makes for the best fit to web development needs.

It is on its own however not complete. The intricacies of the Internet and web media will require extensions to Extreme Programming in order to facilitate effective web development. Borrowing processes and elements from other methodologies, it is proposed a new methodology is created. For the purposes of this analysis it shall be referred to as webXstream, acknowledging the Extreme Programming core with web extensions that it comprises.

Discussed below are pertinent points with respect to implementing Extreme Programming in this web environment.

Discussion of phases

Context analysis

Occurring at the outset of a web project, this process is recommended to determine the business case for proceeding with the effort. The Crystal "Project360" perhaps best embodies this concept; this process attempts to capture 'samples' of key areas to the success of the project (Cockburn 2003). These are discussed below.

Business value and requirements

Involving key stakeholders, this defines what the system should do for its users and their organisation(s). However, rather than a traditional list of key use cases, it is proposed that the Fusebox-style persona / goal modelling technique be employed instead, to gain a higher level though more comprehensive idea of the problem domain (Helms 2003). Working at a more familiar level for the customer, all potential actors are identified as well as their aspirations for use of the system.

This method of design takes an alternative view; it recognises that traditional goal-directed design results in software that seeks to deliver on tasks rather than ideal end goals (Cooper 2002). Features become developed to satisfy these tasks, when there may be alternative ways to achieve the user's objectives that are far simpler or more efficient. While only anecdotal evidence exists as to the effectiveness of this alternative approach, and further research is advisable, considering an issue at a higher level of abstraction than the product itself will intuitively go some way towards a better understanding of the problem domain: the 'big picture.'

Indeed, once this processes is completed the developers will need to analyse these goals, and co-operatively decide upon a model for the requirements - Crystal describes this as a 'domain model'. The translation into terms meaningful to the developers occurs at this stage, aiding project estimation and enabling the technology sample.

Technology viability

At this stage the developers retreat to consider the real-world technological and professional aspects of the domain model. If there are doubts as to the possibility of achieving a successful outcome with any technology, a small experiment (known as a 'spike' in XP) is completed. This should take the form of "the smallest possible programming assignment needed to make it clear that the project is not running down a blind alley." (Cockburn 2003)

Once this is settled and the team is satisfied that

- The technologies can be successfully connected
- The technologies will withstand the loads of the project
- They have sufficient skill to implement the technology

the project may be deemed technologically viable.

Commitment to the project

Only when the business and technological value of the project is established should the team commit to completing the project, usually by means of a formal bid. The bid particulars should be based on the estimates gained from the domain model; further clarification may be required in order to gain sufficient depth of information to estimate effort required for completion.

Requirements modelling

Once the bid has been accepted, modelling of the webXstream system may begin in earnest. This centres on the need to capture what the user seeks to achieve with the end solution.

Lowe (2002) suggests that clients' understanding of their needs "evolve as a system emerges and is utilised" and thus design artefacts "play a crucial role in the development of the clients' understanding.

Iterative approaches such as those in most agile methodologies go some way to assist in the understanding of a system's context and implications - but do not directly assist in developing the client's understanding. This process typically asks a client whether the given design meets their needs, rather than specifically designing means to help explore the problem domain (Lowe et al. 2002).

There are however two aspects of the Fusebox methodology that do seek a better understanding of the 'bigger picture', Persona goals and Wireframes.

Persona goals

eXtreme Programming utilises a storytelling technique to facilitate communication of how a client sees the eventual product working. While this certainly assists in the simplification of requirements interviews - it is an acknowledgement that users are unlikely to possess the knowledge and skills to articulate concrete requirements (Wells 1999) - however it still tends towards thinking in terms of tasks that need to be performed, like activities in a story.

Persona goals force thinking at a higher level, and from different perspectives. This involves considering the system from the point-of-view of anyone who will interact with it - more specifically what these particular users are looking to achieve by using it. In this way, this sort of role-play has potential to assist in understanding of the problem domain.

Wireframes

Fusebox questions the value of textual descriptions and diagrams when representing web systems; at least when directly consulting with the client. For these to work at a layer of abstraction suitable for the developers it has no value to clients, and vice versa.

Instead, the process suggests instantly creating a clickable 'wireframe'. Consisting hyperlinked, plain web pages containing only textual descriptions of what they will eventually look like, it allows both the client and developer to gain an instant understanding of the system structure and its intended user interactions. A simple series of web pages can be quickly mocked up in an on-site client meeting with a HTML authoring tool for instance.

Analogous to the Extreme Programming 'system metaphor', theory suggests that the wireframe is complete when all the identified persona goals can be answered within the potential wireframe interactions. This may not be possible in all cases, so some degree of ambiguity is still acceptable at this stage.

The wireframe serves two purposes - it immediately provides a solution that customers can provide feedback on, and in conjunction with persona goals, forces all parties to consider usage issues from all perspectives - potentially assisting in problem domain development.

Fusebox asserts this early feedback reduces the need for costly changes later on, as constructive feedback can be given upfront. However the web development methodology must be appreciative of changes that may be required due to changes in the business domain and, by extension, business needs with respect to the web project. webXstream should thus anticipate change at any point in development.

Design and construction

An assessment of the existing methodologies shows those which anticipate and accommodate change do so largely through the tool of iterative development. This approach acknowledges that design and construction, as far as software is concerned, cannot be separated as the code forms an intrinsic part of the design itself.

Combining these into finite iterations has the added advantage of completing only portions of the design at any one time - so if changes to other portions are required it does not cause a chain reaction of change through other components.

Partitioning

With a workable wireframe, the developers must partition the model into work packages to facilitate an iterative development process.

In line with Extreme Programming practice, the webXstream work packages should not individually take more than one week to complete. The developers (specifically not management, as per Extreme Programming practice) must estimate the time to completion for each package, then consult with the customer to prioritise the work packages by "business value" and "plausible order of development".

Each work package also has associated with it specific acceptance criteria, which the client defines. This is the benchmark for integration testing once work packages are completed. It should be noted that each work package only describes 'what' needs to be done, rather than 'how' that functionality should be achieved. The design is otherwise entirely in the hands of the developer.

Release planning

Continuing with Extreme Programming practice, the process of release planning may now occur. Here, the remainder of the project is tentatively mapped out. Work packages must be divided between development iterations, which should range from one to three weeks in length. Extreme

Programming dictates the maximum number of work packages that may be completed in an iteration is governed by the project velocity; as mentioned previously this relates to the number of work packages completed in a prior iteration. Continuing this process results in a release plan for the whole project. Given the flexibility in project velocity, clients should expect variation to this plan through future release planning meetings. These may be called by either client or developer as needs arise.

Iteration structure

A key difference between software and web development is the structure of the output product. Whereas generic software may be considered largely homogeneous, web projects tend to be built on layers of different technologies; databases, middleware, front-ends, scripting and so on.

Fusebox accounts for this by initially developing the prototype which eventually becomes the front-end for the finished product. This is in response to two unique web characteristics;

- Significant input is made into the 'look and feel' of a web site; usability is paramount and there is also considerable artistic input into page appearance
- A user is not concerned with any part of the web system but for the front end. Fusebox theory suggests that once this passes acceptance testing the customer will not need further changes - what happens behind the front-end is of no direct concern to them, so developers may construct this however they choose. The only constraint is that the back end has to support all functionality initially mocked up in the front end 'prototype'.

The distinction between front- and back-end development is justified for these purposes. However Fusebox's linear approach to front-end development is perhaps not the most efficient means of completing the task, particularly given the assertion that seventy per cent of the web development process is devoted to this phase (Helms 2003).

The webXstream process considers whether iterative development may assist in this largely graphic design phases as well. To date no evidence could be found to prove or disprove the possibility of productivity improvement, however: " Design artefacts assist in the client's understanding of the problem domain in terms of web systems generally " This allows more constructive feedback earlier in the process

Further research is required to ascertain the effect(s) this proposal may have, positive or otherwise.

Individual iterations

Development iterations should initially focus on developing the front-end in line with Fusebox theory. An iteration begins with developers 'signing up' to complete the selected work packages for that iteration. As Extreme Programming asserts, developers initially create test cases that will form unit tests for each work package as it is completed.

Front-end iterations will obviously require a different testing regime to the later back-end iterations. Usability is the primary measure of front-end effectiveness, so test cases will need to be developed to facilitate measurement of this. Usability testing is an emerging field; it may require as little as a checklist or far more elaborate user testing. The nature of these tests is for the team to decide. Each front-end page has functional requirements, and these will need to be tested as well.

Refactoring and continual integration will be required, as per standard Extreme Programming guidelines.

Further research is required before the complete validity of this process may be ascertained. However, the benefits of paired programming translated into this domain suggest:

- Consistent 'look and feel' remains an important consideration, and a strategic thinking partner can certainly assist in the application of standards to work outputs
- Studies have shown paired programming to assist in programming productivity with a cost

gain of only fifteen per cent (Maurer et al. 2002). Similar gains, with the addition in quality of output could occur with this activity as well. Further research is recommended.

Outcomes

Like Extreme Programming, the webXstream process seeks to establish a product with some business value as soon as the first iteration. This may require skilful partitioning, depending on the environment of the project. If the developed web project is replacing an existing web, release planning may have to cover the interchange between old and new as work products are completed. Iterations will continue, at varying project velocity, until all work packages are complete. Given the consistent testing, communication and consultation, delivery should occur "without surprises" for any party.

Roles within the project team

The traditional Extreme Programming team consists of approximately five to fifteen people, with a significant skew towards developers over managers (Beck 2000). Extreme Programming calls for generic developers who may work on any function of the software system. This is not feasible however for web projects; the required professional mix is too varied. The following is a suggested breakdown of roles in the process, although it will vary by project and budget.

Context analysis

This requires the involvement of most team members, software engineers, security experts, information architects, brand stewards and copywriters as well as account managers. The professional opinions of these team members will assist in estimation of effort, technical plausibility and general understanding of the problem domain - so complete participation of an executive of the client is also necessary.

Requirements modelling

Requiring the same team, although the requirements elaboration can also benefit from the inclusion of actual end users who can assist with persona and goal identification. The information architect would take the lead in wireframe development and work package partitioning, with assistance from software engineers and other technical experts.

Design and construction

Here the graphic designers join the team to develop the front end through the first iterations of the process, guided by the brand stewards and copywriters or art directors. Once the front end is complete, the software engineers and security experts begin development of relevant back-end systems. Account managers facilitate communication between team members and the client throughout.

Discussion of supporting processes

Unlike the traditional heavyweight processes, the agile methodologies that webXstream is based on seek to minimise administrative overheads to the development effort. Two very important supporting processes are however essential to even lightweight regimes; communication and some form of documentation. The discussion begins however with discussion of the use of estimation and metrics in development.

Estimation in software processes

There has been considerable effort put into effort estimation in traditional software development methodologies. Elaborate structures such as the Construction Cost Model and simple calculations like Lines Of Code have been used to guide project planners for some years now (Pressman et al. 2000). It has however been suggested that these measures are critically flawed because development itself is completely unpredictable. There is thus little value in attempting to predict a

phenomenon subject to so many external forces (Maurer et al. 2002).

Rather than have no effect at all, estimation could in fact cause harm. If one is to introduce measured management he or she must get all important factors under measurement (Austin et al. 1996). However, without concrete, definitive measures workers will tend toward improving what metrics are available even if this has the actual result of reducing productivity. Fowler believes this to be the case in software development, where not even the most basic of productivity measurements can be reliably made (Fowler 2003).

This supports the lone 'lightweight' metric used in Extreme Programming, and indeed webXstream. Not used as a measure of progress, the project velocity simply assists in planning for future iterations.

Documentation

A feature of all agile processes is the reduced administrative overhead. This is demonstrated by minimal project documentation; the agile manifesto (Beck et al. 2001) preaches use of the least amount of documentation to allow for effective development.

webXstream project documentation extends to the following:

- Persona goals, used to help the design
- Wireframe, a software artefact that demonstrates structure and functionality of the eventual solution
- Test cases, used to guide development through usability and/or functionality necessary for a work package as required
- The software itself, front-end pages and back-end code.

The strict coding standards that form one of Extreme Programming's practices allow the code to be documentation in itself. Of course, it is up to the team to develop standards that they believe appropriate to the project at hand.

Communication channels

One of the four core values of Extreme Programming is communication; this is another factor that minimises documentation overhead, by utilising the efficiency of spoken interaction between developers, management and clients.

To aid effective communication, a short daily meeting (known as the 'scrum' in Scrum, taken from the Rugby term) is encouraged. The team leader asks three questions of each team member:

- What was done since the last scrum?
- What problems arose?
- What will be done before the next scrum?

No other business is discussed at the scrum; generally the entire process takes no longer than twenty minutes. It serves to simply keep all team members informed of progress project-wide.

Informal interaction between developers is highly encouraged and facilitated through working in close proximity. Constant consultation with the client through an (optional) on-site representative is also highly recommended; if the budget does not allow for this a 'hotline' to an executive client representative may suffice.

Content management

Where relevant, a strategy must be developed to cover the generation of content for the web project, particularly for products such as portals or Intranets.

The plan should define:

- Who shall generate the content: the client, the developers, a third party
- The nature of the content
- How the content will be edited, if necessary
- How the content will be entered: for instance, through a back-end system for client staff? live feeds? ongoing maintenance by developers?

Other aspects and constraints

All agile processes are characterised by the latitude they give to the project team, recognising that only the team knows what works best for them in a given situation. This extends to the process itself; as a part of iteration planning, a review of the process is advised so that group functions can be optimised to a given environment. More formal communication channels may be required, for instance, or perhaps less.

All methodologies are however subject to some form of constraints. Obviously the greatest constraint to effective implementation of elaborate methodologies is budget-oriented. Not all aspects of this radical proposal can be accommodated in the budget of every project, so some discretion is required.

Another potential constraint has been previously discussed, largely concerning the radical nature of an Extreme Programming project. More conservative clients may certainly feel uncomfortable to committing to what is still a poorly quantitatively proven process, despite its intuitive attractiveness. The counter to this argument is that the web itself is a relatively immature media, and with its revolutionary characteristics similarly revolutionary project organisation may be called for.

The webXstream process is presented in Figure 1.

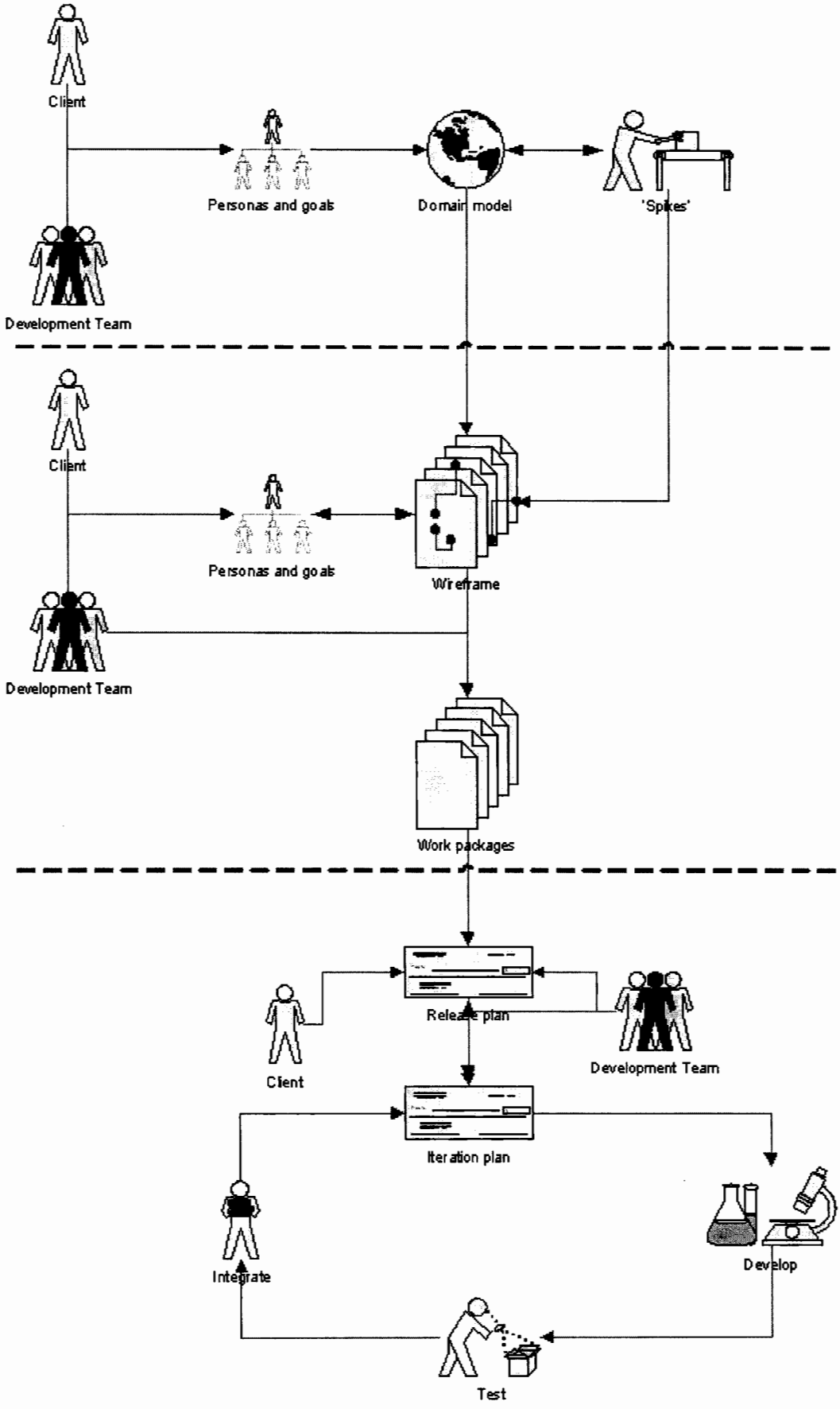

Figure 1 Diagrammatic representation of the webXstream process

Conclusions, verification and recommendations

Up to seventy per cent of web projects are failing to deliver on their intended functionality. There is evidence that:

- Customers do not understand the problem domain within which the solution is to exist, and
- So find it difficult to articulate the system requirements in a meaningful form upfront
- A volatile business environment frequently pressures requirements to change during projects
- What few processes are being used by web developers typically take the form of traditional engineering methodologies, not accustomed to accommodating change in requirements after the design is finalised

There is further empirical evidence that methodology assists in delivering a successful project. However traditional methodologies fail to cater adequately for the structure of the Internet and its technologies. The optimal web development process is better suited to the new-wave 'agile' methodologies in software development; however additional extensions are required to handle the unique characteristics of the web.

The optimal web process

Based largely on the Extreme Programming paradigm, webXstream adds extensions that specifically cater for web development.

Context analysis The business case for the project is established. Close consultation with the client yields Personas of all potential users and their individual goals are role-played. This forms the 'domain model' of the system, from which a technological sample is gathered to ensure the real-world grounding of the technologies required to make the solution happen.

Requirements modelling If the bid based on context analysis is successful, the developers, managers and client collaborate to create a clickable wireframe model of the solution. This series of HTML pages devoid of content but a description of their eventual content is considered complete as a model once all persona goals may be modelled in its use.

Design and construction There is no distinct design phase, the wireframe is simply partitioned into work packages that are prioritised for development, and grouped into iterations. Each work package is developed independently by a pair of developers who choose how to achieve required functionality of their own accord. These are integrated with the other work packages on a frequent basis.

The result is a product that can provide business value by the conclusion of the first iteration. Each iteration adds further value, but the entire release plan can be altered and work packages amended as change requires - for each iteration is partitioned to be completely independent of the others.

Supporting processes Frequent communication is essential in place of the detailed documentation that traditionally accompanies software projects. Daily 'scrums' of no more than twenty minutes keep all team members informed of the whole team's progress, and informal communication between developers, management and clients is facilitated by co-location where possible, or telecommunication otherwise. Strict coding standards and disciplined approaches seek to produce code that essentially documents itself; there is no distinct design documentation.

Verification

The webXstream is verified in terms of optimal characteristics in Table 2.

Table 2: Verifying webXstream

Ideal	Implementation
Usable	Usability is predefined in acceptance tests and implemented in appropriate test harnesses within each front-end development iteration

Timely	Iterative development allows a product of business value to be released after only one iteration; changing business needs accounted for in adaptive iterations. Content management strategy.
Sound technology	Allows developers latitude to select appropriate technologies that are proven plausible (with 'spikes', as necessary)
Requirements aware	Goal-oriented design, utilising user personas. Wireframe developed collaboratively with the client upfront.
Flexible	Iterative design accommodates change by focusing on today's needs as they arise, rather than attempting to forecast potential change before it happens
Documented	Extensive communication regimes replace the need for detailed documentation; code documents itself with application of strict coding standards and pair programming
Allows innovation	Again, complete latitude is given to developers; only constrained by what is required, not how to deliver it.
Economical	Iterative development staggers costs and delivers business value from the first iteration.
Professional mix	Specialised phases acknowledge the roles of various professionals necessary to make the web system as comprehensive as possible.

Further recommendations

Further research is necessary to determine the potential benefits or otherwise that webXstream may bring. There is a small but growing field of research into the effectiveness of agile processes, most notably the 2nd International Workshop on Empirical Evaluation of Agile Processes (EEAP 2003), held at New Orleans, Louisiana, USA earlier this year. However there still lacks repeatable, positive evidence - the example frequently cited to support Extreme Programming itself is the original C3 payroll project at Chrysler which gave rise to the methodology. While it proved to recover from a bad position to a full release in 1997, the project then fell apart two years later. For whichever reasons, this shows that XP is not an infallible process (Fowler 2003).

The other primary field of research required is that into the front-end development of web sites. While intrinsically an art, and so may elude any sort of metric whatsoever no literature could be found relating this practice to any sort of methodology or potential for analysis.

Finally, it too is a growing field; however usability testing is still not well defined enough for application as a test harness to the development of web front-ends. A more consistent definition and measurement scale is necessary to bring discipline to this aspect of design as well.

Acknowledgements

The authors wish to thank Mr Christian Kressig, Mr Adam Good and Mr Aaron Turk of web development firm Tribal DDB Sydney for their kind assistance in reviewing some of the material in this paper. Their industry perspective helped extend some concepts and provoke thought with alternative viewpoints.

References

Abrahamsson, P, Warsta J, Siponen, M and Ronkainen (2003) "New directions on agile methods: a comparative analysis" Proceedings. 25th International Conference on Software Engineering, 3-10 May 2003, pp244 - 254

ADM Inc. (2003) "What is Scrum?" Available online [HREF2]

Austin, R. D., T. R. Lister, et al. (1996). Measuring and Managing Performance in Organisations. New York, Dorset House.

Barkstrom (2003) "The Standard Waterfall Model for Systems Development" NASA EOSDIS Available online [HREF3]

Beck, K. (2000). Extreme Programming Explained: Embrace Change. Reading, MA, Addison-Wesley.

Beck, K., M. Beedle, et al. (2001) "Manifesto for Agile Software Development" Agile Alliance, Available online [HREF4]

Bloch, M. (2003) "eCommerce, Communications and the Global Internet Community" Available online [HREF5]

Burdman, J. (1999). Collaborative Web Development: Strategies and Best Practices for Web Teams. Boston, Addison-Wesley.

Ceglarek, T. (2003) "Modelling web development methodology", Thesis, University of Technology, Sydney

Cockburn, A. (2003). The Game of Programming. A Human-Powered Methodology for Small Teams: Crystal Clear. Reading, MA: 215.

Cooper, A. (2002). "Goal-Directed Design." Cooper.com.

Cyhaniuk, P. (2003) "Getting Started With Personal Computers and the Internet" Flintridge Chamber of Commerce [HREF6]

December, J. (2003) "Web Media Characteristics" December Communications Inc. Available online [HREF7]

Fowler, M. (2003) "The New Methodology" April 2003, Available online [HREF8]

Gormik, D. (2001). IBM Rational Unified Process: Best Practices for Software Development Teams, Rational Software.

Helms, H. (2003) "Fusebox Lifecycle Process (FLiP)" Fusebox.org, Available online [HREF9]

Highsmith, J. (2002). "What is Agile Software Development?" Crosstalk: The Journal of Defense Software Engineering 9(10): 4-9.

Johnson, J., K. D. Boucher, et al. (2001). "Collaborating on Project Success." Software Magazine 7 (2): 15.

Jupin, C. and M. Howell (2003) "The Crystal Family of Methodology" University of Georgia, Available online [HREF10]

Lowe, D. (2000) What makes Web development difficult? Column in WebNet Journal, Vol 1, Issue 2, 2000

Lowe, D. and Henderson-Sellers, B. (2001) Impacts on the development process of differences between web systems and conventional software systems. SSGRR 2001: International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, L'Aquila, Italy, 2001. Available online [HREF11]

Lowe, D. and J. Eklund (2002). Client Needs and the Design Process in Web Projects. The Eleventh International World Wide Web Conference, Honolulu, Hawaii, USA, Internation World Wide Web Conference Committee.

Lowe, D. (2003) Web development process, WebTech'03, Sydney, Australia, July, 2003

Lynch, P. and S. Horton (2002). Web Style Guide: Basic Design Principles for Creating Web Sites, Yale University Press.

Maurer, F. and S. Martel (2002). "Extreme Programming: Rapid Development for Web-Based Applications." IEEE Internet Computing 6(1): 86-90.

Murugesan, S. (2000). Web Engineering for Successful Web Application Development. Asia Pacific Web Conference, Xian, China, AeIMS Research Centre.

Nielsen, J. and D. Norman (2000). "Usability On The Web Isn't A Luxury." Information Week.

Palmer, S. and Felsing, J. (2002) Practical Guide to Feature-Driven Development, Prentice Hall, 2002

Powell, T. A., D. L. Jones, et al. (1998). Web Site Engineering: Beyond Web Page Design, Prentice Hall.

Pressman, R. and D. Ince (2000). Software Engineering: A Practitioner's Approach: European Adaptation. London, McGraw-Hill.

Wells, D. (1999) "What is Extreme Programming?" Available online [HREF12]

Hypertext References

HREF1
http://www.eng.uts.edu.au/
HREF2
http://www.controlchaos.com/
HREF3
http://asd-www.larc.nasa.gov/barkstrom/public/The_Standard_Waterfall_Model_For_Systems_Developm
HREF4
http://www.agilemanifesto.org/
HREF5
http://www.tamingthebeast.net/articles/globalinternetecommerce.htm
HREF6
http://www.lcusd.net/lchs/dclausen/InternetSeminar/powerpoint/internet/sld001.htm
HREF7
http://www.december.com/web/develop/character.html
HREF8
http://www.martinfowler.com/articles/newMethodology.html
HREF9
http://www.fusebox.org/index.cfm?&fuseaction=methodology.steps
HREF10
http://www.arches.uga.edu/~cjupin/
HREF11
http://www.eng.uts.edu.au/~dbl/archive/2001-Low01c.pdf
HREF12
http://www.extremeprogramming.org/what.html

Copyright

form with the conference papers and for the document to be published on mirrors on the World Wide Web.