e-SI whitepaper

Dynamic Adaptive Framework

Embracing the power of change through L.E.A.N Cycle Planning

By Dick Chase and Bill Denneen

Executive Summary

How do you manage technology in dynamic business environments? How do you manage technology projects to take advantage of dynamic business environments? At the end of a long project, how would you answer the questions "Are you where you need to be or are you where you planned to be?"

At e-SI, we believe that the networked business world calls for a new approach to developing and implementing strategic business objectives. Existing strategic and tactical frameworks assume that perfection is approachable and that rigorous attempts to reach perfection are the foundation of success. In most proprietary approaches either speed to market is sacrificed for detail and control, or complexity is avoided until it's too late (complexities are sidestepped or ignored until the project is so far along that they can't be avoided, causing delay, if not outright collapse). In the Networked Economy, however, increasing uncertainty, combined with shortened business cycles, means that excessive control is a path to failure. Success today comes from carefully managing risk. In today's dynamic business environment a new approach is needed. E-SI calls this approach the Dynamic Adaptive Framework, which we implement through LEAN Cycle Planning.

Introduction to E-SI's Dynamic Adaptive Framework

Traditional engineering methodologies are designed to optimize processes by avoiding change as much as possible. However, in a world where the very definition of optimal processes changes frequently, attempting to optimize processes becomes an exercise in futility. In the networked world we need frameworks rather than methodologies. And we need frameworks that promote practice (how to do things) over process (when to do things). We need frameworks that focus on optimizing value obtained and managing risk rather than methodologies that focus on controlling tasks by curtailing change.

e-SI's Dynamic Adaptive Framework (DA Framework) grew out of on-going research in the multi-disciplinary field of Complex Adaptive Systems (CAS). CAS looks at systems from an organic, evolutionary perspective rather than a rules or process based perspective. When adapting CAS to business strategy and (by extension) systems development, we also make an important distinction between complicated systems in the context of static environments (which we believe are amenable to process based approaches) and complex systems in dynamic environments. In the DA Framework, we define and manage complex systems by evaluating and reacting to the levels and form of variation, interaction and selection inherent in those systems.

For example, a successful web community requires a wide variety of participants, along with a wide variety of ways for those participants to interact with each other individually and with the community as a whole. By carefully evaluating how community members contribute to and utilize the community, we can observe a natural selection of community elements by participants, which we can capitalize on to further enhance the value of the community to its members. While it may be possible to determine in advance what would be valuable to members of our proposed community at the start, the very nature of a dynamic community makes it impossible to know what members will find valuable later on, as that value is defined only within the context of members in the community and their interaction at a given point of time.

In short, the utility of the DA framework can be viewed as an application of discipline to a fundamental truism in modern business: If you don't know where you are going - any road will get you there.

LEAN Cycle Planning

LEAN Cycle planning, in turn, is e-SI's tactical expression of the DA Framework for implementing business systems in dynamic environments - a framework created to avoid what we call "The Planning Trap."

The Planning Trap

Of all the changes wrought by the Internet, perhaps the most far-reaching is the destruction of the Plan-Build paradigm. E-Businesses live in a world where change is fast and the unexpected is the norm. Planning and designing in long cycles, according to expected realities, cannot provide a sustainable competitive advantage to e-Businesses.

In static or slowly changing environments, project methodologies that force the practitioner to anticipate and thus avoid change *is* the most efficient and practical approach. Such methodologies have proven very successful in projects where inputs and outputs can be well known in advance. Examples of such design and build undertakings include software that automates existing tasks such as accounting or payroll systems, construction projects such as roads or bridges, or medical equipment such as respirators or heart monitors. In such ventures either the number of variables, which would necessitate design modifications, remain low or at least relatively stable over time, or maintaining constancy of inputs and outputs do not affect the overall value of the planned output. In all of these examples, avoiding change is an achievable and desirable goal.

Exploring the specific example of a software payroll system, we can observe that the number of taxes that may affect payroll payments are fairly stable within given political boundaries (which, in turn, are also reasonably stable) over time. Tax rates may change (and designers can anticipate that they will change and confidently known that the rates would not likely exceed 100%), but number of taxes is unlikely to change over fairly long stretches of time (and can be efficiently handled by newly designed versions of the software every few years).

The underlying commonality of projects such as this lay in the value, if not necessity, to the end result of avoiding or ignoring change. Utilizing traditional engineering methodologies in such a project, a project manager would create a highly detailed project plan and manage all goals with a firm eye towards preventing "scope creep" (i.e. change). If change is needed, due either to external changes (the U.S. government passes a universal healthcare plan that involves a new payroll tax, affecting the design of the payroll application), or internal changes (the designers missed that Rhode Island has unemployment tax withholding as well as income tax withholding), the project manager elevates the issues as they arise to a level where they are negotiated (either the design, implementation plan and/or cost is adjusted to absorb the change, or the change is put off for a possible future version).

Building business on or adapting business to the Web (or building any system that interacts with people or other dynamic entities), of course, is different. Six months into building a carefully planned new Internet music service, you may look up to find millions of people trading music files amongst themselves in a way you never anticipated. Do you continue

building an obsolete business or go back to the proverbial drawing board, if resources even allow?

When implementing systems for dynamic business environments there are **always** issues of change that either the project manager needs to elevate for negotiation, or that the project sponsor needs to introduce in order to obtain optimal value from the implementation. The latter is arguably even more important than the former: When using traditional engineering methodologies, we avoid new ideas because negotiation is an exception in the process.

Since, by definition, change is constant in dynamic environments, change management must also be a constant in our implementation practices. As we'll see, in Lean Cycle Planning, change management is not part of the process; it *is* the process. Rather than negotiating change reactively (and thus painfully) on an as needed basis (thus as infrequently as humanly possible), Lean Cycle Planning makes recognizing and negotiating change a central element of the process. In LEAN Cycle Planning, negotiation, rather than implying failure (poor requirements gathering or design), reflects reality (change happens).

In today's dynamic business environment, ignoring change is perilous, fighting it foolhardy. The most rational course of action is to embrace change.

Embracing Change = Managing Risk

Vulnerability to unknown future changes, i.e., risk, is a fact of life in business. In fairly static systems it is safest to anticipate possible changes and plan ahead, but this strategy will backfire in a more dynamic environment, locking you into a course of action and inhibiting your ability to adapt. Successful e-Businesses manage risk – they don't eliminate it.

With e-SI's LEAN Cycle Planning we embrace change by articulating a strategic vision; focusing on goals and results instead of processes and inputs; posting and recognizing boundaries while understanding that the boundaries can change; deciding which trends to follow, which to ignore, which to start.

Table 1: Methodologies vs. Frameworks

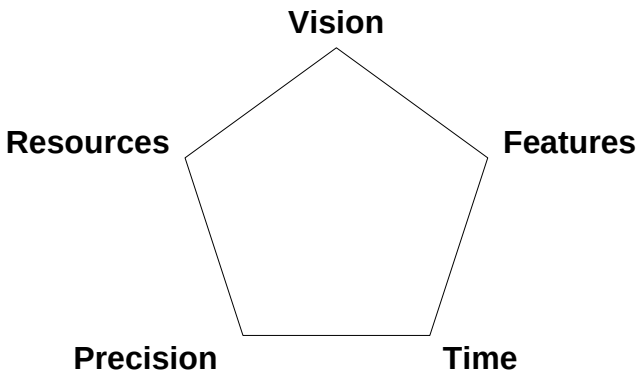
	Industrial Age Methodologies	Network Era Frameworks
Risk mitigation strategy	Predict and Plan (linear progression)	Learn and Adapt (cyclical feedback)
Driving force	Strategic Plan	Strategic Vision
Design-and-Build process	Discrete stages with hand-off points	Collaborative revision with evaluation points
Focus	Process	Results
Boundaries	Fixed center – tend to expand in dynamic environments	Movable center – Size remains stable in dynamic environments
Ideal environment	Static	Dynamic

When the people in your organization understand the vision and are empowered to reach for it, change works to your advantage. Human perception and collaboration make it possible.

Embracing Change ≠ Recklessness

Guided by the overall vision, negotiations occur within defined boundaries of features, time, precision and resources.

Figure 1: Boundaries



We all know that the relationship between these elements centers on compromise. But we choose to ignore the effect of that compromise in dynamic environments. Historically, in the practice of managing projects, project teams either convince themselves that they can change one of these elements without affecting the other, or they put all of their strength into combating any change to any element. Through apparently facile adaptation of traditional engineering methodologies consultants, managers, and developers provided the illusion that they can control these elements. The truth is, however, that such control on a practical level is simply impossible in truly dynamic environments. However, while we can't control them, we can *manage* them to maximize the value obtained through an undertaking. We do this in LEAN Cycle Planning by defining a Project Boundary Matrix[1]:

Figure 2: Project Boundary Matrix

	Priority		
Project Factors	Primary	Secondary	Good Enough
Scope (Features)			
Schedule			
Precision			
Resources			

[1] The Lean Cycle Planning boundary matrix is adapted from the "Product Mission Profile Matrix" as described in James A. Highsmith's *Adaptive Software development.*

- A project has one and only one primary factor. When evaluating changes, all other factors defer to satisfying this core objective.
- A project has one and only one secondary objective. This objective defers only to the primary dimension
- All remaining factors are evaluated on a "Good Enough" basis. Goals are set for these factors, with acceptable agreed upon margins for flexibility (generally +/- 10%-20%). Optimization, however, is sacrificed for other factors.

This becomes the foundation for negotiation. The constitution, if you will. Consider a project with the following parameter definition:

Figure 3: Sample Project Boundary Matrix

	Priority		
Project Factors	Primary	Secondary	Good Enough
Scope (Features)	X		
Schedule		X	
Precision			Limit non-invasive bugs (e.g. works with IE 5.0, w/ bugs that don't affect core functionality in lesser versions)
Resources			$250,000 (+/- $35,000)

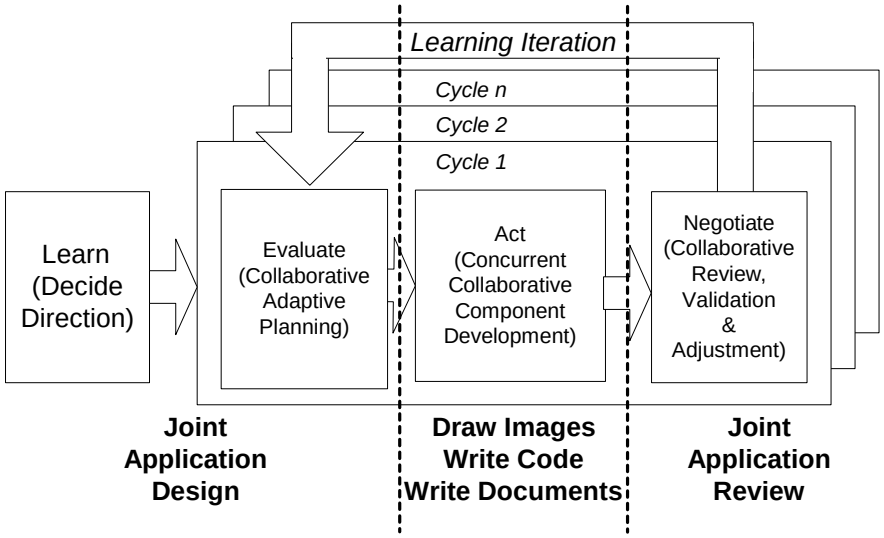
Possible change issues and their negotiated resolution for this project could include:

- If fixing a bug affects the schedule or ability to complete other features, it is delayed for a future release phase.
- If adding a feature that enhances core functionality requires more money or more time, then schedule, precision and resources are adjusted to allow for the enhancement.
- If the system can be delivered sooner by adding another developer (resources) or by *not* fixing a minor display problem on Netscape 4.0, then add the developer and put off fixing the Netscape browser issue for the next version.

Learn Evaluate Act Negotiate (LEAN) Cycles

To replace the Plan-Build paradigm, e-SI's LEAN Framework integrates Vision and Reality in tight collaborative cycles, perceiving changes in the environment as they occur and incorporating them into the design in real time.

Figure 4: L.E.A.N. Cycles



Learn

Take what's known and decide direction. This is input into the cycle. If the start of a project, it is the strategy work, that serves as the basis for creating the project definition. In the next cycle it is the result of the negotiation phase. Most important is that it is done in collaboration with client in workshop format. Ideally the workshop takes place in-person, though a videoconference is acceptable. Teleconferences are ***not*** acceptable substitutes.

Evaluate

Figure out how to achieve the objective. Evaluate opportunities for action. This phase reflects most closely traditional project methodologies. Tasks are determined and assigned.

Act

This too resembles traditional framework. Actual coding, writing of documents or creation of images is done. The difference is that it is ***Concurrent Collaborative Component Development***. While there is a focus on specific components in a given cycle, ***all components are worked on***. The essential Project Management skill is making sure balance is maintained.

Negotiate

This phase is the key. It is not a status meeting. It is not a presentation of documents or slides (unless that is ultimate output). It is a demonstration of the product as a whole, in whatever state it is in. Key skill is facilitating the conversation.

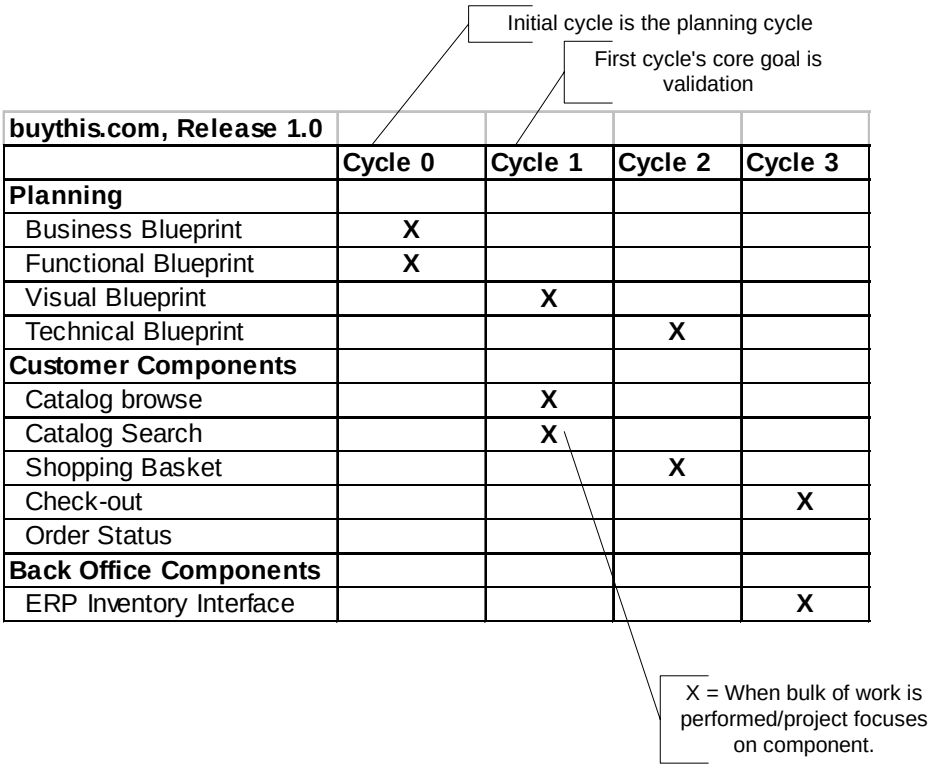
Negotiation revolves around the entire application, not just one or two components, regardless of the state of completion of individual components. Most change requests for

'approved' parts of a project arise because the components were evaluated for approval by themselves outside of their final context. Reviewing individual components by themselves also results in clients 'hiding' (or our not discovering until its 'too late') changes/second thoughts about non-reviewed components.

Mapping The Cycles

These cycles are mapped out in a Time-Boxed Cycle Map[2]. This is similar to a traditional project plan, the difference being that the plan maps out deliverables rather than tasks. Components are identified for focus in each cycle. This does not mean that those components will be completed in that cycle. Rather, it means that those components will be in reviewable form, within the context of the entire application.

Figure 5: Time-Boxed Cycle Map



buythis.com, Release 1.0				
	Cycle 0	Cycle 1	Cycle 2	Cycle 3
Planning				
Business Blueprint	X			
Functional Blueprint	X			
Visual Blueprint		X		
Technical Blueprint			X	
Customer Components				
Catalog browse		X		
Catalog Search		X		
Shopping Basket			X	
Check-out				X
Order Status				
Back Office Components				
ERP Inventory Interface				X

Risk drives the number and length of cycles. The greater the risk for a component (un-proven technology, un-proven interface, unproven model, etc), the shorter its cycle

[2] The Lean Cycle Planning cycle map is adapted from time-boxed cycles as described in Adaptive Software Development.

Cycles should not be less than one week and no more than 10 weeks. Longer projects can have longer cycles, though adjusted according to the dynamism of the business or technical environment (the more dynamic, the shorter the cycle). For projects of six months or more entailing a significant amount of risk, plan for short cycles up-front, with longer cycles later. In addition to managing risk efficiently, shorter cycles up front serve to demonstrate that progress is being made early on in a project.

Collaborate, Collaborate, Collaborate

The key to success in concurrent collaborative development is seamless real-time collaboration among all members of a project team. Collaboration means more than regular communication and call reports, status reports and sharing artifacts. It means sharing and, more importantly actively discussing, the vision of what the project will produce – the "projections and models of what the components will look like in the end."[3] Fortunately, the internet and "groupware" such as Notes, eRooms and Intraspect and "p2p" software such as Groove makes such collaboration not just realistic, but relatively painless[4]. With Net based collaboration tools members of a team can continuously review the activity of other team members that could affect them, and notify team members of activity that may affect them.

But technology solves only part of the problem. To truly facilitate collaboration from the human perspective, contributing to discussions about the project must be given a high priority as a part of everyone's job responsibility, with individuals who actively contribute or facilitate collaboration duly rewarded. Ideally, recognizing that sharing information is not an innate activity for humans, a project team should also include one individual whose primary role is to actively gather and disseminate information and foster discussion.

Exploration and Exploitation

One of the benefits of concurrent component development separated by negotiation events is that it allows for both exploration (evaluating new ideas of potential import ant value) and exploitation (executing proven ideas) within the context of the overall deliverable. A healthy balance of exploration and exploitation is a vital part of any healthy complex adaptive system, though with an inherent direct trade off between the two[5].

Thus, one of the most important skills of both the developer and project manager is the artful balancing of exploration and exploitation such that new methods and ideas are explored, while successful ones are exploited in a way that achieves project goals. Identifying the proper balance objectives is most efficiently done during the negotiation phase of each cycle.

---

[3] Axelrod & Cohen, Harnessing Complexity, p. 86
[4] To enable such collaboration, e-SI utilizes software from Intraspect that allows for all members of a project to have full access to all elements of a project.
[5] Axelrod & Cohen, Harnessing Complexity, p. 44

Negotiate to Learn

Indeed, in addition to resolving issues of change, the negotiation can be used to enhance knowledge that can be used later on in the current project and in future projects. Because of the tremendous value obtained during negotiation, it is vital that the negotiation phase be regarded with the utmost sincerity. It cannot be skimped upon. It cannot be informal. At a minimum negotiation should utilize one full day at cycle end for any cycle in any project. It is reasonable to expect negotiation to occupy two to three days in a normal project. In large projects (especially where major issues trade-off decision need to be made early on), expect negotiation to take up to a week.

Each cycle ends in a review session focused on a prototype. Wordy documents, flip charts, and power point slides are not allowed! Business sponsors don't understand dataflows, architectures or function points. And they don't have to. They understand the important stuff, like whether something has value to them. Prototypes allow discussion to occur in languages and in *contexts* that all parties understand.

Facilitating negotiation between developers, project managers, and business sponsors, requires a certain level of objectivity that is usually not present in members of a project team. Successful negotiation sessions require a trained facilitator not directly involved in the project outside of their role as facilitator.

Anatomy of a Project

This all sounds good, but how do you actually do this?

In LEAN Cycle Planning, projects are defined by five types project definition documents:

1 - Project Declaration
2 - Project Architecture
3 - Project Model
4 - Project Component Definition
5 - Project Briefing

We'll explore the details of each document form individually below. At a high level, the Project Declaration and Architecture could be viewed together as the "constitution" for a given project. These are generally developed at the start of a project and remain unaltered throughout the life of a project. Indeed, significant changes to the content of either of these documents generally means that the nature of the project itself is being altered and should be re-evaluated as a whole. The Project Model, Briefing and Component Definitions are modified according to the results of each negotiation cycle.

For very large undertakings, creation of project definition documents should be spread across multiple LEAN cycles, with the negotiation phase revolving around the contents of the documents themselves.

The Project Briefing is a simple one or two page summary of the project. This allows someone unfamiliar with a project to gain an understanding of a project - its goals and its current state - quickly, without having to wade through four inches of technical specifications (OK, for the 21st century, make that four megabytes of technical specifications).

Project Declaration

The project definition defines what we want to accomplish by undertaking the project, why the goal is important, what we're willing to commit towards reaching this goal, and how we'll know when we've gotten there.

Vision

Project Vision Statement – Define the project vision in "elevator test" format[6]:

For [target customer] **who** [need or opportunity addressed] **the** [product name] **is a** [product category] **that** [key benefit of product]. **Unlike** [main competitive alternative]. **Our product** [primary differentiator].

For example:

> **For** active investors **who** need to limit their risk or exposure to the volatility of markets, **the** moneymaximizer.com site **is a** web based investment tool and calculator **that** sizes orders for investment vehicles to an investors stated level of acceptable risk. **Unlike** analysis tools that let you analyze past trends, **our product** allows you to maximize future gains by minimizing current losses.

Background - what brought about this vision?

Business Functional Objectives - Business benefits that project addresses. The problems solved and/or opportunities created.

People

Put a human face on the project for everyone involved. If technology solutions are about people, we must understand the people involved. This includes:

- Executive sponsor - the person behind the project & why
- Internal constituencies - who within the organization will be affected by what's being created & how.

[6] Moore, Crossing the Chasm.

- External constituencies – who outside of the organization will be affected by what's being created & how will they/should they be affected. External constituencies can include customers, partners, or both.

Boundaries

Scope, Time, Resource and Precision targets are defined and prioritized in the project boundary matrix. Note that these should be executive-level targets (basic functionality, schedule, budget, and reliability goals).

Tangents

What the project does not intend to accomplish. These are paths that the project could take, but that, at this time, we don't wish to take. This helps focus design and execution, and helps frame discussions during the negotiation phase of each cycle.

Success Metrics

Our definition success and how we will recognize it.

Risks

Potential events that would affect a ***successful outcome***[7]:

- Internal risks (e.g. the staff won't accept it, resources get reallocated by others etc.)
- External Risks (e.g. a competitor gets there first, the market shifts abruptly etc.)

Project Architecture

This is a definition of the mechanics of the project. An individual reading the Project Architecture should come away with an understanding of how the project is organized.

Vision & Boundaries

Repeat the vision statement and boundary matrix from the Project Declaration.

[7] The words "successful outcome" cannot be overemphasized here. Risk does not include events that could change the course of a project, only events that could result in failure to meet identified success metrics or crossing the boundary limits identified in the boundary matrix.

Roles & Responsibilities

Who will be involved, on both developer and client side & what their individual responsibilities are. Named people if known, but names aren't required. This should also include any specific skill requirements known.

Technical Performance Objectives

Performance targets, how to measure them and why.

Prerequisite, Dependent Projects and External Deadlines

What's needed from other projects or activities, what other projects or activities require output from this project. Also notes of any external deadlines (e.g. must be ready by annual industry trade show).

Constraints

Hard limitations such as staff or technologies (e.g. must use Oracle on NT, only one junior IT employee will be available to support application, etc.).

Assumptions

What's being assumed/taken for granted. (e.g. Skill of client staff, partner staff availability, costs of tools etc.)

Risks

Risks faced by project w/r/t resources and abilities

- Internal (e.g. skilled staff leave the organization, infrastructure won't handle finale product, etc.)
- External (e.g. Can't find needed technology, vendor stops supporting product used by a prerequisite project, etc.)

Project Model

The Project Model defines the project components and cycle map. It is the functional equivalent of the traditional project plan, differing only in that the plan focuses on deliverable rather than tasks. The model is discussed and adjusted as needed during the negotiation phase of each cycle.

Vision & Boundaries

Repeat the vision statement and boundary matrix from the Project Declaration.

Components

Describe project components. These descriptions should be short and geared towards putting a component into context of the project as opposed to describing how the component will work or how it will be built.

Risk

Evaluate the risks associated with each components. At e-SI we have created a scoring matrix for evaluating the risk factors for each component, though the actual risk assessment for each component will be left to the judgment of the project manager. In large projects, the whole project model itself should be the subject of its own cycle.

Cycle Map

The components laid-out in a time box.

Resource Requirements

- Hardware/Software Requirements
- Staffing/Skill requirements
- Estimated Costs
- Significant deadlines

Current Cycle objective

25 word statement (in the same form as the project vision statement), with list of success factors.

Project Component Definition

Each component should have its own detailed definition. Component definitions together can be thought of as the equivalent of the traditional functional specification, especially in the sense that the development team works from the component definitions in determining the tasks they need to perform to deliver individual components. Component definition are very different from functional specifications, however, as the components are defined as distinct deliverables in discreet documents, as opposed to collection of functions in a thick project specification document that includes everything and the kitchen sink.

Definition

- Form of component (e.g. a widget or chunk of application functionality; a document like a help file, business plan, a web page, a graphic, etc)
- Function of component - what should the component do - this is where we have detailed functionality. Interface components should have work-flow definitions. Non-visual technology components should include programming interfaces.
- Technology used - what's used to create the component.

Prerequisites & Dependencies

What's needed from other components, projects or activities, what other component, projects or activities require output from this component.

Roles & Responsibilities

What roles must be filled to achieve the desired results and what the roles are responsible for delivering to the project and other team members. Specific team members should be named here.

Component State

A component can have one of four states at any given time:

- Outline (conceptual definition)
- Model (Review working model)
- Negotiated (reviewed w/ negotiated changes documented)
- Approved (completed, ready for release/integration with final product)

Current Cycle Objective

Objective for this component during current cycle. The objective should be brief (25 words or less), with a bulleted list of success factors. From this short objective statement, a developer, designer, manager or business sponsor should recognize what the result of the current cycle should be w/r/t this component.

Cycle Summaries

Summary of status, progress, state, issues… from the end of each cycle.

Project Briefing

A summary of one to two pages (when printed on letter size paper) for

quick reference. People are unlikely to read a long document unless absolutely necessary. But you want anyone who's the least bit interested in a project (like a salesperson selling something similar or a PM starting something similar) to be able to quickly get what's important.

- Vision Statement & Boundary Matrix (from Declaration)
- People (summarized from declaration)
- Roles & Responsibilities (from Project Architecture, w/ specific team members identified)
- Component cycle map (from Project Model, w/ current cycle and component states)
- Current Cycle Objective (from Project Model)

---

e-SI

www.esionweb.com

Bibliography

Axelrod, Robert and Cohen, Michael D. *Harnessing Complexity: Organizational Implications of a Scientific Frontier*. New York: The Free Press, 1999.

Highsmith, James A. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. New York: Dorset House Publishing, 1999.

Schrage, Michael. *Serious Play: How the World's Best Companies Simulate to Innovate*. Boston: Harvard Business School Press, 1999.

Sawhney, Mohan. *Strategy In The New World of Business*. Presentation to "Roundtable Group E-Commerce Bootcamp", 2000.

Kaner, Sam et al. *Facilitator's Guide to Participatory Decision-Making*. Gabriola Island: New Society Publishers, 1996.

Farrell, Winslow. *How Hits Happen: Forecasting Predictability in a Chaotic Marketplace*. New York: HarperBusiness, 1998.

Senge, Peter M. *The Fifth Discipline: The Art & Practice of the Learning Organization*. New York: Doubleday, 1990.

Haeckel, Stephan H. *Adaptive Enterprise: Creating and Leading Sense-and-Respond Organizations*. Boston: Harvard Business School Press, 1999.