



# Building Mission Critical Systems in Internet Time



A Report From the Front-Lines

*Andrew C. Lim*  
*Managing Partner*  
*NetSource Partners, LLC*  
*alim@nspartners.com*

*Joe N. Milligan*  
*Managing Partner*  
*NetSource Partners, LLC*  
*jmilligan@nspartners.com*

*August 12, 1998*

## Abstract

In the heat of battle, those survivors on the front-line amass a body of knowledge that increase their probability of survival and success of the overall campaign. As with all combative endeavors, taking time to analyze the success or failure of strategic and tactical plans is instructive to those engaged in the conflict. In cyberspace, where fierce competitive battles are being waged, there is plenty to learn about rapidly deploying mission critical applications and services to support these campaigns in this hostile environment. This paper outlines some of those lessons.

## Mission Profile

*"... to build the worlds fastest growing Internet service in less than 12 months to AT&T quality standards"*

During the early spring of 1996, a division at AT&T Laboratories was tapped to develop AT&T's first Internet service. At that time, they were engaged in building and operating an on-line service called PersonaLink. They were already considered combat veterans of the cyberspace wars, so they were a logical choice for the next campaign.

Their new mission was to build the world's fastest growing consumer Internet service in less than a year to AT&T quality standards. This new service was to be the foundation of AT&T's entry into the consumer Internet space and to showcase the "new" AT&T's market responsiveness. It was critical to show how AT&T could adapt to new markets after divesting itself of both Lucent and NCR, and the new WorldNet Services offering was to be one of the first examples. By having such a clear mission, it gave them a clear picture of what they were all working towards and provided a mechanism for unifying their efforts.

In cyberspace, the rate of innovation and growth is exponential. Adapting to this time compressed frontier, while living up to a tradition of quality would require adoption of leading edge development processes. The obvious question of how to accomplish this mission in "Internet time" was addressed by team of development engineers. Some of the lessons learned from these engineers and how top-level management can capitalize on them are discussed in this paper.

## Development Model

In order to have any hope of accomplishing their mission, they needed a model of how to approach their development efforts. This model represents their battle plan and will provide the foundation for organizing both the work and the development staff.

*"... the model represents their battle plan ..."*

Instead of using a standard waterfall development model, which relied on completing each phase of development (requirements, architecture, design, etc.) before moving on to the next phase, they chose to use an incremental and iterative model. This model is based on doing a little of each phase and moving onto the next one. Once you have completed a cycle, you pause and evaluate how you did, what has changed in the environment, and how to proceed for the next cycle. As you iterate through the development lifecycle, you incrementally add new features. This lifecycle is shown in Figure 1.

## Building Mission Critical Systems in Internet Time

*A Report From the Front-Lines*

They define the development phases along traditional lines, but the processes used to execute the phases are significantly streamlined and adapted to the application being developed. The general definitions for each phase are:

*Business Plans* - include marketing projections, operational objectives, budgeting constraints, partnership agreements, customer care requirements, and other functional area business plans. These plans provide the context in which you are planning to deploy your application. Given that this context is continually changing, you need to review the plans at least once per iteration through the development lifecycle. Plans that can affect your application include changes in marketing projections for the take rate of your application. For example, the original marketing figures were increased based on market feedback and early demand. These increased market projections then impacted the subsequent phases of the lifecycle.

*Requirements* - is a specification of what your application will be designed to do. This specification should start out with simple short descriptions for each feature. The feature specification should evolve with successive iterations through the development lifecycle to include feature specifications for mainline, support, and performance features. The quantity of feature descriptions should be scaled to the amount of work you are planning to accomplish for the current iteration of the development lifecycle. You should also include brief end-user scenario descriptions of your applications, as your requirements document matures.

*Architecture* - describes the computer system, software, and networking components used to implement your application or service. The architecture should begin with a simple component diagram and progress to a more detailed discussion of how all the components will interact to support the documented end-user scenarios.

*Design* - adds lower level details to your architecture. For system and network components, the design will begin to specify connectivity, and configuration specifications. For software, the design will specify how the software components will be put together and how they will interact. The design of all components will evolve with iterations of the development lifecycle.

*Development* - includes the software coding, network and system configuration, and component level testing. The development should proceed to the point of building the necessary pieces to begin integration testing. The scope of the development should be limited to the features targeted for the current iteration of the lifecycle.

*Testing* - integrates all the current components in a test environment that allows system level verification of the current features. The test plans, test cases, and automated test tools will evolve with each lifecycle iteration.

*Deployment* - involves moving the applications and services from the lab to either a pre-production environment, which may exist as part of the production environment. This phase will allow you to develop procedures for configuration, migration, support, and operations of

*"... each completed iteration of the lifecycle should be a meaningful subset of your complete service or application ..."*

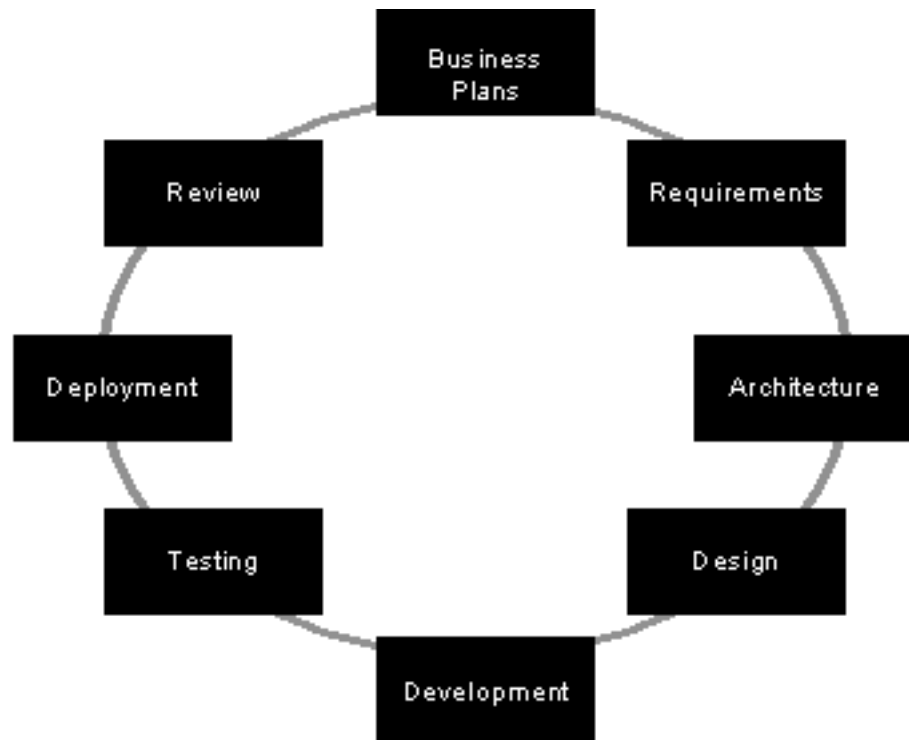
## Building Mission Critical Systems in Internet Time

*A Report From the Front-Lines*

your new application or service. As with the other phases, you will evolve these procedures with successive iterations.

*Review* - requires you to pause and evaluate. This step is vital to increasing both the application/service quality and speed for deployment. You will need to examine your activities during the previous development lifecycle from two perspectives. First, you will need to look at "what" you did. Did you accomplish enough work? Did you do too little? Did your delivery meet the requirements? Second, you need to examine how you did it? Are there better methods and process for performing the work at each phase? How can you streamline the each phase? After asking yourself these questions, you can then begin to plan how to tackle the next iteration of the development lifecycle.

**Figure 1. Incremental & Iterative Lifecycle**



For this incremental and iterative development model, an iteration of the lifecycle should be a meaningful subset of your complete service or application. In addition, it is worthwhile to schedule reviews, either informal or formal, after every development phase. These reviews will give you a good in process assessment of the development products (i.e. requirements, architecture, design, etc.).

## Live Fire Exercises

Now that they have a battle plan, they need to practice execution. Just as no general or admiral would go in to battle without exercising and training their troops, no development manager should ever consider deploying

## Building Mission Critical Systems in Internet Time

*A Report From the Front-Lines*

mission critical systems without practicing the entire development lifecycle with their staff. These development lifecycle iterations are "live fire" exercises for the development staff, their managers, the business managers, and the operations staff.

"... lifecycle iterations are "live fire" exercises ..."

As with all practice and training exercises, the value they provide is allowing the participants to learn from the mistakes they inevitably make. This learning process will allow them to perform better and deliver more with successive iterations. You will see that the early iterations will be slower to complete, because all the involved personnel will be formulating the roles/responsibilities, defining the processes, and learning how to work with each other.

You should note that these "live fire" exercises also benefit the marketing and product managers, since they will get previews of the mission critical system allowing them to provide input before the next practice round. This business input is valuable, especially if the business environment and requirements are rapidly changing.

## Organization and Logistics

All successful campaigns devote a significant amount to organizational and logistical issues. This investment in people and process issues are critical to improving both the performances of the staff and the products they produce.

For each phase of the lifecycle, your organization will adopt a variety of processes to accomplish their development tasks. In order to improve these organizations and processes, Worldnet used ideas from both corporate re-engineering from Michael Hammer's *Re-engineering the Corporation*, and continuous process improvement from Total Quality Management principles. By applying these principles and adopting the lifecycle model previously presented, they were able to achieve:

- Up to an 83% reduction in development lifecycle times.
- Run multiple parallel development teams to further reduce overall development schedules to 11 months.
- Deployment of early pilot releases to get business and customer feedback.

"... reduction in cycle time is possible if the entire team can practice together and apply re-engineering and quality improvement principles ..."

At Worldnet, they were organized around the process of delivering new services to the field. For each new service such as e-mail, they would have a development team that was responsible for the end-to-end process of moving the service from concept to production. These "server" teams relied on support and infrastructure teams to supply them with services (e.g. Engineering and testing), but the "server" teams had the responsibility for insuring the entire development lifecycle process was followed. This organizational model explicitly focuses on fitting the organization to the development lifecycle and process, instead of force fitting a development process into an existing organizational structure.

By organizing around the development and deployment process, the "server" teams could apply re-engineering and quality techniques to improve their performance and output. For example, a "server" team responsible deploying the tools and systems used to operate the Worldnet Service took over two months for their first iteration through the lifecycle. By removing steps in the development, testing, deployment

## Building Mission Critical Systems in Internet Time

*A Report From the Front-Lines*

phases, and by streamlining processes in requirements and design phases, they were able to reduce their development intervals to two weeks for effectively supporting most new services. This 83% reduction in cycle time is possible if the entire team can practice together and apply re-engineering and quality improvement principles.

In addition to organizing around development and deployment processes, they replicated this approach across several "server" teams to carry out development in parallel. This application of the development lifecycle across parallel development teams allowed further shortening of the development schedule. These parallel development efforts came together during integration testing cycles, which subsequently resulted in pilot releases of the applications and service. Once a pilot release was deployed, they would begin planning for the next iteration through the lifecycle and the production of the next pilot release. Pilot releases were made available to limited usage, much like beta releases of software.

There is a final point about organizing around the development lifecycle. It is essential to have one or more individuals who are responsible for guiding the team(s) through the development lifecycle. These process owners and guides could be a project managers, lead engineers, or development managers. From an organizational perspective, these people should be highly valued because they will accumulate critical intellectual assets about how to efficiently move your organization through the lifecycle. Much as generals rely on their field officers for their experience, CIO's and senior management should rely on these process guides to help improve the development lifecycle.

## Model Adaptation Under Fire

Another critical requirement for quickly building mission critical systems is how you modify the development model as you iterate through the lifecycle. This model adaptation comes in two forms. The first type of adaptation is based on continually applying process re-engineering and process improvement techniques to each development phase. It is not good enough to set up processes and tools for the first iteration and assume that you are done. By continually improving each development phase with successive iterations, you will reduce your cycle time, improve your application quality, and build a better development organization.

The first type of adaptation focuses on accomplishing each phase more efficiently, while the second type of adaptation has to do with adapting the lifecycle model itself. There is nothing sacred about the different phases of the incremental and iterative lifecycle presented in this paper. You will find that different development projects will eliminate or significantly reduce products of each development phase. The important idea is to evaluate the impact of eliminating or reducing the products of each phase. Simply put, if you do not need a development phase or product of a development phase, then do not do it.

"... improving your development phases with each iteration will reduce your cycle time ..."

In all mission critical applications today, development organizations rely on third party software components to build their systems. Worldnet and other AT&T on-line services were no different. For those portions of the Worldnet Service being purchased or re-used, there was no need for the design and development phases. In practice, the design and development phases were reduced to gaining an understanding of how the purchased software was organized (design phase) and how to

## Building Mission Critical Systems in Internet Time

*A Report From the Front-Lines*

configure and package it (development phase). Understanding how your major subsystems of your mission critical application fit into the development lifecycle allow you to make intelligent decisions of how to adapt the lifecycle. By practicing your development lifecycle by iterating through it several times, you can then gain feedback on your lifecycle adaptations.

### Lessons for the Generals and Admirals

The lessons to be learned from the front-lines of the Internet are straight forward, but take significant effort and focus to implement. These lessons are:

- **CLEAR MISSION** - clearly define your development objective.
- **USE ITERATIVE & INCREMENTAL MODEL** - use an iterative and incremental development lifecycle model to allow your organization to practice developing and deploying your application or service.
- **ORGANIZE AROUND LIFECYCLE AND PROCESSES** - build your organization around the lifecycle and development processes. Do not try to force fit a streamlined development process into current organizational structures. Rely and retain those people who gain the experience of leading an organization through the lifecycle and empower them to improve it.
- **APPLY RE-ENGINEERING AND PROCESS IMPROVEMENT** - have your lifecycle development teams devote significant effort to re-engineering and improving each development phase and the lifecycle with each iteration.
- **USE PARALLEL DEVELOPMENT TEAMS** - replicate the development teams work in parallel and synchronize at fixed points during the lifecycle.
- **ADAPT YOUR MODEL** - adapt the lifecycle model itself to fit your development needs. The model should be modified to account for the type of development and integration you do to build your mission critical system.

From the lessons above, you should be able to increase your probability of deploying mission critical applications in Internet time and with required levels of quality.