# Faster, Easier and Cheaper Software Development: Is It Possible?

*Using Application Lifecycle Management to improve your software development process*

*By Beth Schechner*

*Elementool*



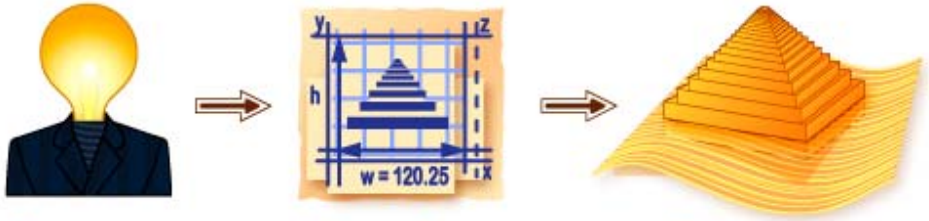
www.elementool.com

# Faster, Easier and Cheaper Software Development: Is It Possible?

*Using Application Lifecycle Management to improve your software development process*

## What is Application Lifecycle Management?

An application's lifecycle is characterized by a three significant phases. It begins with a **concept**; the idea to create a particular piece of software that performs a certain function. Once the idea has been put into action, the application is in **development**. The piece of software is being built, developed, and will soon be available for use doing whatever it is that the concept originally intended. Finally, after the application has been developed (which is a whole other discussion), it's ready for **deployment**. With most applications, once they are deployed and out in the world, issues and bugs arise and ideas for updates are generated. When this happens, your software goes back to development (and all that entails), to start over again.




The Phases of Application Lifecycle Management

The process of managing this cycle is called Application Lifecycle Management, or ALM. ALM links business management (looking out for time, money, function, market and practicality) with software development (developing, testing, tracking and fixing). This is all made possible by a suite of tools and applications that both facilitate and integrate the two groups. A common collection of ALM tools include requirements management, testing, issue tracking, and time tracking. ALM tools should encourage communication between all teams involved in both the business and the software development sides.

They should also act as a reference for business and IT, to check that both are on the same page throughout the software lifecycle.

ALM is applicable, and useful, to all different development approaches, including agile, iterative or waterfall development.

For color, let's imagine a business (yours) that wants to create and develop a new software application such as a multi-player game.  Why should you bother implementing ALM?

## Why ALM is Important

As a whole, Application Lifecycle Management increases productivity and lets your teams share best practices for development and deployment.  ALM allows your developers to focus on business requirements and maintain consistent quality, so that the final game will meet the needs and the expectations that you created, for all end-users

ALM streamlines development, and also speeds the whole process up; decreasing the time it takes to develop your new game and bring it to market.  It maximizes your investment in labor and technology by making sure that each step in the development process is completed as efficiently as possible.

Each step within your application's lifecycle is important in its own way. Similarly each step of ALM has its own value. For instance, testing makes sure that you're not only maintaining the quality of your software, but improving the game with each step.

Software development can be illustrated, or thought of, as series of stages.  Each stage is a mini-lifecycle, and each stage ties into the previous step and into the next step. Every stage of the software lifecycle contains some requirements definition, some design, some development, and some testing. Once the Software Development Lifecycle (SDLC) process for the first version of your game is complete, your multi-

player game will be deployed to your (possibly pre-established) audience of gamers. In most instances, deployment doesn't mark the end of development for that piece of software. Indeed, most applications require occasional updates and fixes.

It's common within IT and the software development industry to equate ALM with the SDLC. However, this idea can sometimes be too confining; ALM is much more than just the development phase of an application's lifecycle. An application's lifecycle, as mentioned above, begins when the idea of an application is conceived and ends only when the entire application (your multi-player video game) is retired and no longer in use. ALM gives guidance and structure throughout this entire period. Below, we'll talk about the different tools and ways you can use ALM to accomplish this.

## Common ALM Steps

*Requirements Management* is the first step of ALM. Before you can do anything else, you need to identify, prioritize and agree on application (and project) requirements. Once these basic steps are handled, they need to be communicated back to all relevant parties including both development and business management. Requirements management is continuous throughout a project; your finished game hasn't met your business goals if it doesn't meet the original, and agreed upon, requirements!

*Quality Assurance (QA) Testing and Test Case Management*, alongside *Issue Tracking*, usually comes next. *Testing* is the stage where you need to manage and track individual test cases. This is where tools really come in handy. It can take several tests to determine whether or not your game is working correctly (your multi-player game, for instance, likely has several hundred different functions, and each individual function must be tested), a Test Case Management tool can be particularly useful in tracking and managing your future, ongoing and completed test cases. A Test Case is a set of conditions under which the tester can determine whether or not the application is working as originally planned, or conceived. This is arguably one of the more important steps in ALM. Testing will make sure each individual function

and feature of your game is working correctly, and if it's not, then an issue is logged and you go back to *Issue Tracking*. Testing is ***traditionally*** handled by a quality assurance team, whose job is to make sure that the best possible product is deployed at the end of the SDLC.

By using a Test Case Management tool, you can keep track of your completed tests, current tests and all the tests that you need to run sometime in the future or delegate to someone else. Similarly to an *Issue Tracking* Tool, reporting functions help tie testing back to business processes, keeping everyone up-to-date on whether or not your game is on track to meet the scheduled release date.


*Issue Tracking* or *Bug Tracking* takes place during development and throughout the rest of the time that you continue to market and support your game. *Issue Tracking*, which ideally runs concurrently with *Testing,* is the stage where you manage and maintain a list of issues or bugs within your game. Although it is by no means the ideal situation, customers do occasionally discover bugs once a product has been released and these bugs, of course, need to be addressed.

Issues or bugs are an inevitable part of software development, no matter what you call them. Having an easy and efficient way to keep track of them will help make sure your game is released to your market

without them (or with as few as humanly possible). The purpose of an Issue Tracking tool is to help the QA and the development teams keep track of these bugs. Many tracking systems will let end-users submit bugs themselves, in other cases bugs can only be submitted by those within the developing organization. Which option you choose depends on your business and processes.

The main benefit of using an issue tracking system is to give a clear overview of development needs including bugs, improvements and fixes, and the current status of each. In your Issue Tracking system, you can assign priority and/or severity to each issue, further clarifying the urgency of every one. You can also keep a backlog and history trail for each new release or product; however you choose to present it, maintaining an accessible issue history keeps your development team from having to reinvent the wheel for each new release.

Again, reporting ties everything back into the overall ALM process, ensuring that business processes and development are on the same page and also working at the same time.

*Time Management* is another vital part of ALM, and again helps to determine project length as well as efficiency. While tracking bugs and testing features is necessary to developing a great and well received game, if your game isn't finished within the project timeline, or falls over budget, then it's not meeting the goals you originally set back in the concept phase.

Focusing on these processes improves efficiency, which leads directly to increased quality over the entire application lifecycle.

Over the years, Application Lifecycle Management has expanded outside of the original tools offered for coding, to provide larger management and oversight over the application lifecycle and smaller development lifecycle. The best of the newer ALM tools offer a common interface and a platform for developers to communicate and share experiences throughout the different development steps.

**ALM in 2010 (and Beyond)**

As software development techniques, and of course technology, change and improve so do the ALM tools that providers are offering. A number of vendors today have started to provide ALM tools that are integrated and work together to streamline the entire process. It does seem obvious to offer a range of management tools which all work together and can be accessed from one dashboard. Additionally, and equally obvious seeming, vendors are beginning to package Application Lifecycle Management tools and more broad-based project management tools together. After all, an application lifecycle is a project, and development teams can certainly benefit from the same organizational and time management tools made available to larger audiences.

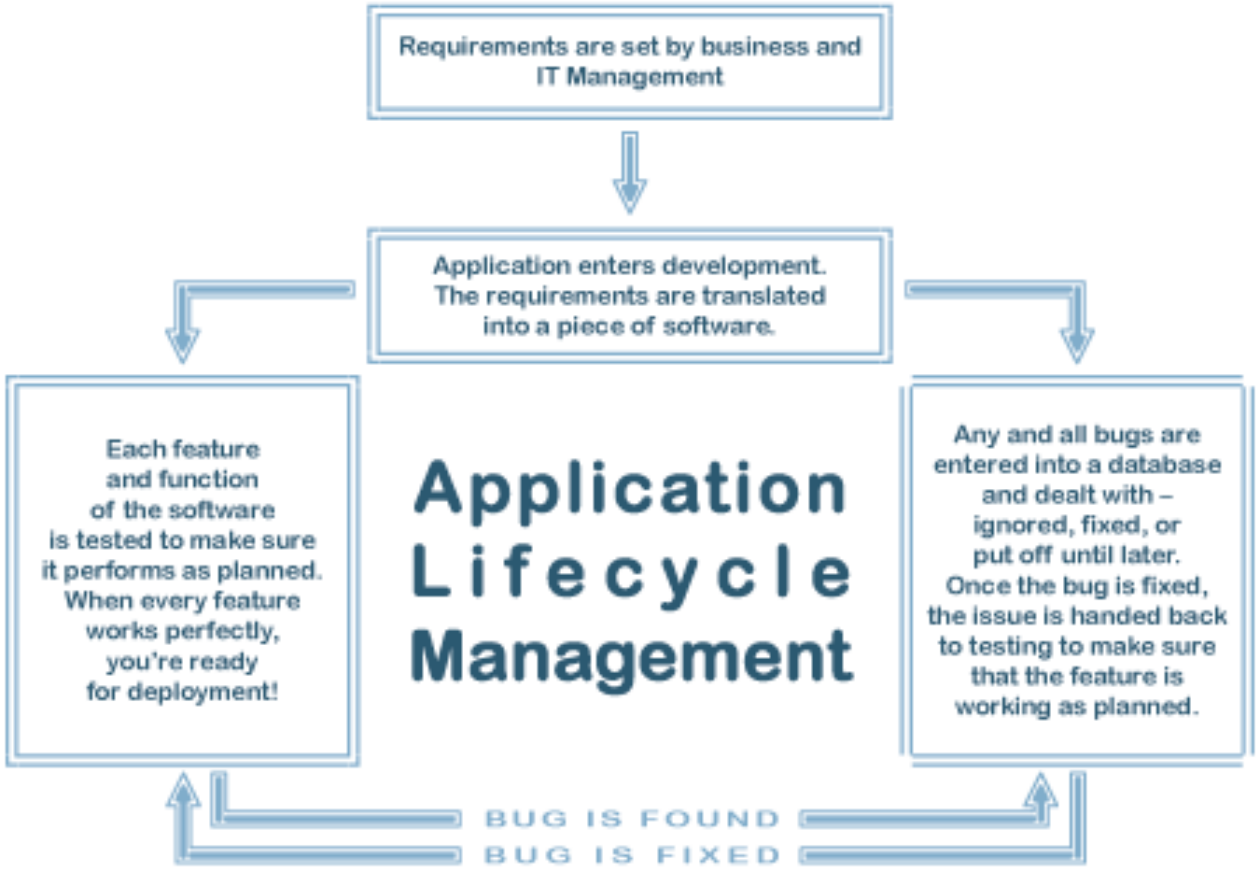
All of these advances also have the effect of shortening the development lifecycle. The days of 18 to 24 month software development projects are over, and the timeframe is beginning to look more like 3 months. This too, creates the need for one universal dashboard for project management, rather than a whole slew of a la carte applications from a bevy of different providers.

The rise in popularity of methods like agile development, also challenge the traditional norms of ALM. Rather than separate roles for your developer, manager and tester, tasks and titles are becoming interchangeable. With the classic Waterfall approach to ALM, it was fine for information and history to be segregated in different tools. However, as roles and titles are becoming more flexible, more flexible tools are needed. The barriers between QA and development, the day-to-day activities of developers, QA testers and other ALM team members are overlapping as project management becomes more accessible and the project lifecycle shrinks.

Logically, this means that there needs to be more communication! Share notes, use forums, integrate chat functions, have meetings – just keeping the lines of communication open will decrease the odds of fatal bugs or malfunctions slipping through the cracks or worse yet, release of a product that doesn't fit with business goals. This after all, is the entire point of ALM. By fostering communication, you're making sure that your developers, testers and YOU are all on the same page and misunderstood requirements don't result in unplanned or unwanted features. Not only are you saving time by keeping lines of communication open, but in the case of the misunderstood requirements, you're saving money by not having to go back and re-develop pieces of your game.

Additionally, communicating business priorities ensures that your development team spends their time on high priority tasks first; solving bottlenecks and preventing team members from spending time on low priority items that could cause a delay in the project.

Personally, an ALM solution should help an organization make connections across all the stages of the application's lifecycle, from the initial conception to retirement. So it only makes sense that project management tools should work together with development tools, which in turn should integrate with the tools used for operations. Each stage is connected, so for ALM to really be useful, the tools should be connected as well. The whole point of ALM is to link the work of the tester, the developer, and the business manager in order to make sure that the software being developed meets business needs and goals.




www.elementool.com

## In Conclusion

With this whole eBook on ALM, why it is important (and useful!) and why tools make the whole thing easier, it needs to be mentioned that tools can streamline workflows, but support from management and a good team of developers and testers is required to make the workflow work! You can come up with a great idea for an multi-player video game, and you can put in place all the recommended tools and processes to build it, but if you don't have a great team doing it than you probably won't end up with a great game.

However with a great idea, a dedicated team, an attentive business management team and Application Lifecycle Management, you CAN build a great application quickly, easily and cheaply.