SYSTEM ANALYSIS & DESIGN NOTE

Source: <https://www.tutorialspoint.com/system_analysis_and_design/system_analysis_and_design_planning.htm>

Following are the seven phases of the SDLC

**1. Planning**

This is the first phase in the systems development process. It identifies whether or not there is the need for a new system to achieve a business"s strategic objectives. This is a preliminary plan (or a feasibility study) for a company"s business initiative to acquire the resources to build on an infrastructure to modify or improve a service. The company might be trying to meet or exceed expectations for their employees, customers and stakeholders too. The purpose of this step is to find out the scope of the problem and determine solutions. Resources, costs, time, benefits and other items should be considered at this stage.

**2. Systems Analysis and Requirements**

The second phase is where businesses will work on the source of their problem or the need for a change. In the event of a problem, possible solutions are submitted and analyzed to identify the best fit for the ultimate goal(s) of the project. This is where teams consider the functional requirements of the project or solution. It is also where system analysis takes place—or analyzing the needs of the end users to ensure the new system can meet their expectations. Systems analysis is vital in determining what a business"s needs are, as well as how they can be met, who will be responsible for individual pieces of the project, and what sort of timeline should be expected.

There are several tools businesses can use that are specific to the second phase. They include:

* CASE (Computer Aided Systems/Software Engineering)
* Requirements gathering
* Structured analysis

**3. Systems Design**

The third phase describes, in detail, the necessary specifications, features and operations that will satisfy the functional requirements of the proposed system which will be in place. This is the step for end users to discuss and determine their specific business information needs for the proposed system. It"s during this phase that they will consider the essential components (hardware and/or software) structure (networking capabilities), processing and procedures for the system to accomplish its objectives.

**4. Development**

The fourth phase is when the real work begins—in particular, when a programmer, network engineer and/or database developer are brought on to do the major work on the project. This work includes using a flow chart to ensure that the process of the system is properly organized. The development phase marks the end of the initial section of the process. Additionally, this phase signifies the start of production. The development stage is also characterized by instillation and change. Focusing on training can be a huge benefit during this phase.

**5. Integration and Testing**

The fifth phase involves systems integration and system testing (of programs and procedures)—normally carried out by a Quality Assurance (QA) professional—to determine if the proposed design meets the initial set of business goals. Testing may be repeated, specifically to check for errors, bugs and interoperability. This testing will be performed until the end user finds it acceptable. Another part of this phase is verification and validation, both of which will help ensure the program"s successful completion.

**6. Implementation**

The sixth phase is when the majority of the code for the program is written. Additionally, this phase involves the actual installation of the newly-developed system. This step puts the project into production by moving the data and components from the old system and placing them in the new system via a direct cutover. While this can be a risky (and complicated) move, the cutover typically happens during off-peak hours, thus minimizing the risk. Both system analysts and end-users should now see the realization of the project that has implemented changes.

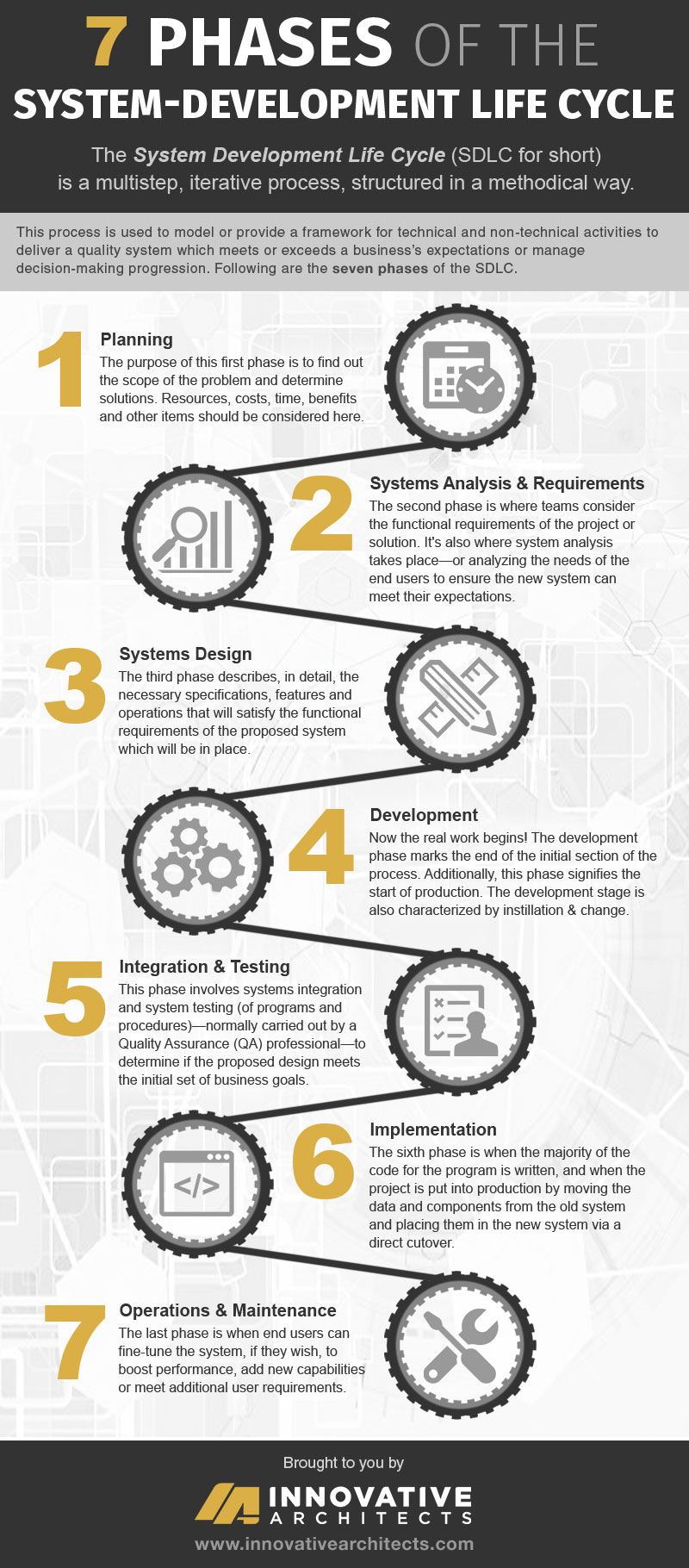
**7. Operations and Maintenance**

The seventh and final phase involves maintenance and regular required updates. This step is when end users can fine-tune the system, if they wish, to boost performance, add new capabilities or meet additional user requirements.

**Importance of the SDLC**

If a business determines a change is needed during any phase of the SDLC, the company might have to proceed through all the above life cycle phases again. The life cycle approach of any project is a time-consuming process. Even though some steps are more difficult than others, none are to be overlooked. An oversight could prevent the entire system from functioning as planned.

Systems development specialists at Innovative Architects possess extensive experience in managing these type of projects. If you have a situation at your organization and you think a customized software solution may be what you need, [contact us](https://www.innovativearchitects.com/ContactUs.aspx) today. [Consultants](https://www.innovativearchitects.com/Microsoft-Consultants.aspx) at Innovative Architects will be able to quickly guide you through each of these steps, ensuring you can have your new system online as soon as possible.



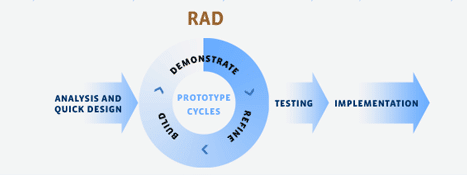
Rapid Application Development

## What is rapid application development?

Rapid application development is a form of Agile [software development methodology](https://blog.capterra.com/what-is-sdlc/). Unlike Waterfall methods, RAD emphasizes working software and user feedback over strict planning and requirements recording.

In other words, RAD is less talk, more action. Oh, and testing. Lots and lots of testing.

While RAD de-emphasizes strict planning, there are still a handful of steps or phases each development project goes through when using the rapid application development methodology, which we’ll discuss below.



### 1. Figure out the requirements

Every piece of software or app is built for a reason. RAD starts with figuring out what your project is supposed to accomplish.

Get your users, developers, and designers into a room (or on a conference call) to discuss the purpose of the system you’re about to build and test. Once you’ve figured out the system requirements, you should also come up with an estimated project timeline. And, of course, you’ll need to figure out how you can create a system within the confines of your budget.

### 2. Build prototypes, on the double!

RAD stands for rapid application development, so it shouldn’t be surprising that your team will start working on building functional models right away. Once you’ve got the requirements, deadline, and budget figured out, your engineers and designers will create and improve upon working prototypes until you’re ready to unveil your finished product.

### 3. Get user feedback

With RAD, your users should already be familiar with most, if not all, parts of your finished product before you announce your project’s completion. Like most Agile methodologies, RAD calls for ongoing collaboration between your team and users in order to create a high-quality system. Your users will be the ones providing feedback so that you can tweak and improve your prototypes, creating the best possible finished product.

### 4. Do it again!

You’ll repeat steps two and three until you feel like your project is done or until you have all working parts assembled together to meet a client’s requirements.

### 5. Test, test, test

You’re not ready to pronounce a project complete until you’ve made sure that it works the way it’s supposed to. Run your system through different scenarios and make sure all of the moving parts work together to accomplish the system’s goal.

In addition to your engineers testing and retesting your code, this will involve more user testing before you take your system live.

### 6. Go present your system!

Once you’ve got a working system, pat yourself on the back! That’s not an official part of RAD, but your team has worked hard to get your project done rapidly. They deserve the congratulations. Now you’re finished… until the next update is required, of course.

## What are the advantages of RAD?

James Martin came up with rapid application development because he thought it worked better than other SDLC methodologies that were around in the 1990s. Below are some of the advantages of using RAD over more traditional methodologies such as Waterfall.

Breaking a large project down into smaller tasks is an advantage in two ways. When developing a large, complicated piece of software, it can help you to naturally form more specialized teams. For example, you can get your expert engineers working on a tricky piece of code while more junior team members take on a simpler part of the project.

That first advantage leads into the second one: you can create small wins for your team to motivate them while they’re working towards a large and difficult goal. Finishing one simple piece of your system up front gives you more hands on deck to help with more complicated pieces. That means that everyone can give more focus to that complicated piece since they’re not preoccupied with thinking about all the smaller pieces they have to perfect before creating a finished product. And who doesn’t want a more focused team when it comes to more difficult problems?

By presenting working pieces, your team can put them all together at the end and, voila, you’re done! And since they start creating working prototypes right after they get out of the requirements meeting, they should constantly have something to show for their hard work. While you won’t have a final product until you’ve stitched all your prototypes together, you’ll always have finished pieces to show your clients.

Always having something to show your client means you can get their feedback and quickly implement any changes that need to be made before they’re added to your finished product.

Ideally, this means that you never have a client who, during your final presentation, says something like, “This isn’t what I expected.” Users are heavily involved in the development process, so their vision will shine through in your finished product.

## When should your team use RAD?

Unfortunately, James Martin didn’t solve the mystery of application development when he came up with the RAD framework. RAD doesn’t work for every project and, like any organizational methodology, shouldn’t be used indiscriminately. But it does work well in a few specific instances, which we’ll discuss below.

It’s all in the name here. If you’re working on a tight deadline, you should consider using RAD, as you’ll produce a working system more quickly than with more traditional methods such as Waterfall. At the very least, you can produce functioning parts of systems that might be “good enough for now” solutions for clients who needed software yesterday.

Since user feedback is one of the advantages of using RAD, you can’t skip out on that step in the process. RAD depends on constant improvements suggested by users, so you’ll need to make sure they’ll be available to collaborate with your development team throughout the process.

Providing users for testing might be part of negotiating your timeline and budget with your client. The good thing is, since you’re getting your project done quickly, users theoretically won’t need to clear much time in their schedules to test your software iterations.

It should come as no surprise that creating a high-quality application quickly requires skill and precision. That means hiring talented designers and engineers. And that means paying them their deserved higher salaries. If you skip out on hiring a skilled team, you might find that you get what you pay for, as work done hastily but shoddily isn’t likely to meet your clients’ expectations.

## Alternatives to rapid application development

There are lots of ways to develop software outside of the RAD system. But we’re going to focus on the two that address the main criticisms of RAD.

If your criticism of RAD is that it’s not structured enough, you can always go back to the [Waterfall methodology](https://airbrake.io/blog/sdlc/waterfall-model). As I mentioned, RAD was created in response to the inadequacies of Waterfall for software development.

While Waterfall isn’t a go-to for most modern developers, it can work well, for example, if you’re working in a very structured environment. So if your client is a large corporation, Waterfall might be the more natural choice for getting your project done smoothly.

If your issue is that RAD is too expensive, [lean software development](http://www.disciplinedagiledelivery.com/lean-principles/) might be your best bet. Lean development methodologies focus specifically on reducing waste throughout a project. That can be done by encouraging the scrapping of unneeded features early on in development or placing your team’s autonomy over your client’s needs.

While you can still use part of RAD or other methodologies when implementing lean development, the focus of lean is working on a thin budget while still creating a high-quality project.

## Is your team RAD?

It’s been a couple decades since RAD first entered the software development scene. Now I want to hear from you. Does the rapid application development process still work for your team? Or have you found other methods to help organize your team?

DSDM

DSDM is vendor-independent, covers the entire lifecycle of a project and provides best practice guidance for on-time, in-budget delivery of projects, with proven scalability to address projects of all sizes and for any business sector.

DSDM advocates the use of several proven practices, including:

* Facilitated Workshops
* Modelling and Iterative Development
* MoSCoW Prioritisation
* Time boxing

DSDM is designed to be easily tailored and used in conjunction with traditional methods such as PRINCE2® or to complement other Agile approaches such as Scrum.