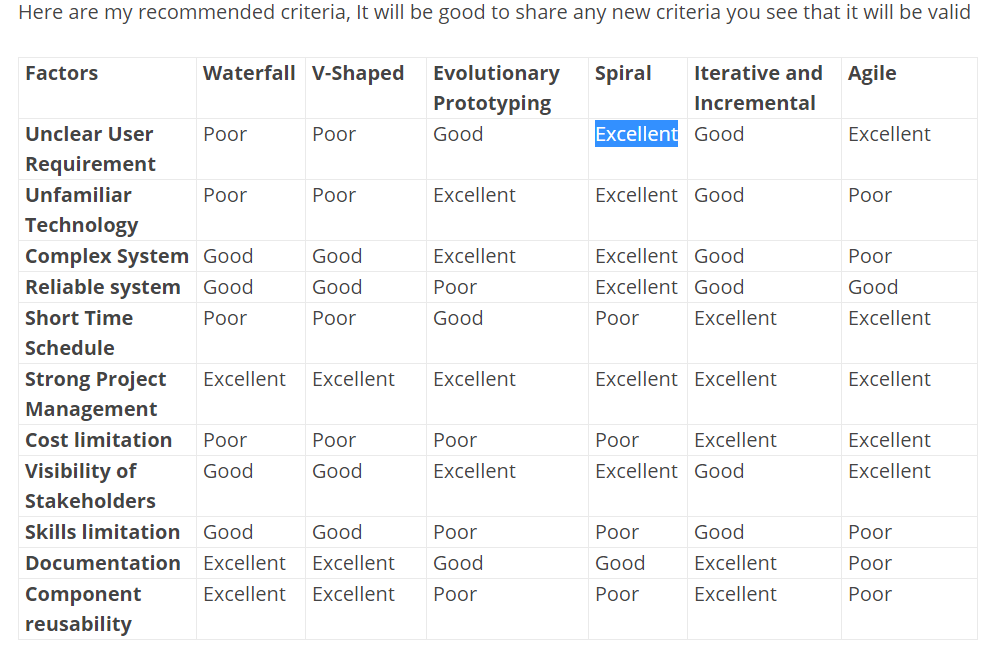


STEP 3: Define the criteria

Some of the selection criteria or arguments that you may use to select an SDLC are:

* Is the SDLC suitable for the size of our team and their skills?
* Is the SDLC suitable for the selected technology we use for implementing the solution?
* Is the SDLC suitable for client and stakeholders concerns and priorities?
* Is the SDLC suitable for the geographical situation (distributed team)?
* Is the SDLC suitable for the size and complexity of our software?
* Is the SDLC suitable for the type of projects we do?
* Is the SDLC suitable for our software engineering capability?



Definition of 'Waterfall Model'

**Definition:**The waterfall model is a classical model used in system development life cycle to create a system with a linear and sequential approach. It is termed as waterfall because the model develops systematically from one phase to another in a downward fashion. This model is divided into different phases and the output of one phase is used as the input of the next phase. Every phase has to be completed before the next phase starts and there is no overlapping of the phases.  
  
**Description:**The sequential phases described in the Waterfall model are:  
  
1. Requirement Gathering- All possible requirements are captured in product requirement documents.  
  
2. Analysis Read - the requirement and based on analysis define the schemas, models and business rules.  
  
3. System Design -- Based on analysis design the software architecture.  
  
4. Implementation Development of the software in the small units with functional testing.  
  
5. Integration and Testing Integrating of each unit developed in previous phase and post integration test the entire system for any faults.  
  
6. Deployment of system - Make the product live on production environment after all functional and nonfunctional testing completed.  
  
7. Maintenance Fixing issues and release new version with the issue patches as required.  
  
Advantages: 1. Easy to use, simple and understandable, 2. Easy to manage as each phase has specific outputs and review process, 3. Clearly-defined stages, 4. Works well for smaller projects where requirements are very clear, 5. Process and output of each phase are clearly mentioned in the document.  
  
Disadvantages: 1. It doesn’t allow much reflection or revision. When the product is in testing phase, it is very difficult to go back and change something which is left during the requirement analysis phase.  
  
2. Risk and uncertainty are high.  
  
3. Not advisable for complex and object-oriented projects.  
  
4. Changing requirements can’t be accommodated in any phase.  
  
5. As testing is done at a later phase. So, there is a chance that challenges and risks at earlier phases are not identified.

## Advantages & Disadvantage of Prototyping process model

  Tuesday, November 24, 2009 2:13 AM -   [SEblogs](https://www.iotap.com/blog/authorid/50/seblogs) -   [Quality Assurance & Quality Control](https://www.iotap.com/blog/categoryid/34/default)-   [2 Comments](https://www.iotap.com/blog/entryid/124/advantages-disadvantage-of-prototyping-process-model#Comments)

The Prototyping Model is a systems development method (SDM) in which a prototype (an early approximation of a final system or product) is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed. Its used commonly in design and development of physical things usually when large scale building construction or manufacturing is involved. This model also works with Software and is best in scenarios where not all of the project requirements are known in detail ahead of time. It is an iterative, trial-and-error process that takes place between the developers and the users. its useful when trying to implement a [Dynamics 365](https://www.iotap.com/services/crm-contact-us) project or a [SharePoint project](https://www.iotap.com/services/sharepoint-services) when you want to demonstrate the interaction with a system. 

There are several steps in Prototyping a System:

1. Identify as many requirements in as much detail as possible. This usually involves interviewing a number of users representing all the departments or aspects of the existing system.
2. A preliminary design is created for the new system
3. A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
4. The users thoroughly evaluate the first prototype, noting its strengths and weaknesses, what needs to be added, and what should to be removed. The developer collects and analyzes the remarks from the users.
5. The first prototype is modified, based on the comments supplied by the users, and a second prototype of the new system is constructed.
6. The second prototype is evaluated in the same manner as was the first prototype.
7. The preceding steps are iterated as many times as necessary, until the users are satisfied that the prototype represents the final product desired.
8. The final system is constructed, based on the final prototype.
9. The final system is thoroughly evaluated and tested. Routine maintenance is carried out on a continuing basis to prevent large-scale failures and to minimize downtime.

Advantages of Prototyping process model

**Reduced time and costs:** Prototyping can improve the quality of requirements and specifications provided to developers. Because changes cost exponentially more to implement as they are detected later in development, the early determination of what the user really wants can result in faster and less expensive software.

**Improved and increased user involvement**: Prototyping requires user involvement and allows them to see and interact with a prototype allowing them to provide better and more complete feedback and specifications. The presence of the prototype being examined by the user prevents many misunderstandings and miscommunications that occur when each side believe the other understands what they said. Since users know the [problem domain](http://en.wikipedia.org/wiki/Problem_domain) better than anyone on the development team does, increased interaction can result in final product that has greater tangible and intangible quality. The final product is more likely to satisfy the users desire for look, feel and performance.

Disadvantages of Prototyping process model

**Insufficient analysis**: The focus on a limited prototype can distract developers from properly analyzing the complete project. This can lead to overlooking better solutions, preparation of incomplete specifications or the conversion of limited prototypes into poorly engineered final projects that are hard to [maintain](http://en.wikipedia.org/wiki/Software_maintenance). Further, since a prototype is limited in functionality it may not scale well if the prototype is used as the basis of a final deliverable, which may not be noticed if developers are too focused on building a prototype as a model.

**User confusion of prototype and finished system**: Users can begin to think that a prototype, intended to be thrown away, is actually a final system that merely needs to be finished or polished. (They are, for example, often unaware of the effort needed to add error-checking and security features which a prototype may not have.) This can lead them to expect the prototype to accurately model the performance of the final system when this is not the intent of the developers. Users can also become attached to features that were included in a prototype for consideration and then removed from the specification for a final system. If users are able to require all proposed features be included in the final system this can lead to conflict.

**Developer misunderstanding of user objectives**: Developers may assume that users share their objectives (e.g. to deliver core functionality on time and within budget), without understanding wider commercial issues. For example, user representatives attending[Dynamics 365](https://www.iotap.com/solutions/customer-relationship-management)events may have seen demonstrations without being told that the specific feature demands additional coding and often requires more resources and time. Users might believe they can expect the same functional and performance characteristics in their production environments,  whereas developers might think this is [feature creep](http://en.wikipedia.org/wiki/Feature_creep) because they have made assumptions about the extent of user requirements. 

**Developer attachment to prototype**: Developers can also become attached to prototypes they have spent a great deal of effort producing; this can lead to problems like attempting to convert a limited prototype into a final system when it does not have an appropriate underlying architecture. (This may suggest that throwaway prototyping, rather than evolutionary prototyping, should be used.)

**Excessive development time of the prototype**: A key property to prototyping is the fact that it is supposed to be done quickly. If the developers lose sight of this fact, they very well may try to develop a prototype that is too complex. When the prototype is thrown away the precisely developed requirements that it provides may not yield a sufficient increase in productivity to make up for the time spent developing the prototype. Users can become stuck in debates over details of the prototype, holding up the development team and delaying the final product.

**Expense of implementing prototyping**: the start up costs for building a development team focused on prototyping may be high. Many companies have development methodologies in place, and changing them can mean retraining, retooling, or both. Many companies tend to just jump into the prototyping without bothering to retrain their workers as much as they should.

These considerations are part of project planning and implementation steps. Talk to us about your Cloud Migration projects to Azure, Upgrades or implementation projects for [SharePoint](https://www.iotap.com/services/sharepoint-services) and [Dynamics CRM](https://www.iotap.com/services/crm-contact-us) where we walk you through a implementation plan and methodology.

*Rapid application development* (RAD) is an incremental software development process

model that emphasizes an extremely short development cycle. The RAD model is a

“high-speed” adaptation of the linear sequential model in which rapid development

is achieved by using component-based construction. If requirements are well understood

and project scope is constrained, the RAD process enables a development team

to create a “fully functional system” within very short time periods (e.g., 60 to 90 days)

[MAR91]. Used primarily for information systems applications, the RAD approach

encompasses the following phases [KER94]:

**Business modeling.** The information flow among business functions is modeled in

a way that answers the following questions: What information drives the business

process? What information is generated? Who generates it? Where does the information

go? Who processes it? Business modeling is described in more detail in Chapter

10.

**Data modeling.** The information flow defined as part of the business modeling phase

is refined into a set of data objects that are needed to support the business. The char-

*Resist pressure to*

*extend a rough*

*prototype into a*

*production product.*

*Quality almost always*

*suffers as a result.*

CHAPTER 2 THE PROCESS

acteristics (called *attributes*) of each object are identified and the relationships between

these objects defined. Data modeling is considered in Chapter 12.

**Process modeling.** The data objects defined in the data modeling phase are transformed

to achieve the information flow necessary to implement a business function.

Processing descriptions are created for adding, modifying, deleting, or retrieving a

data object.

**Application generation.** RAD assumes the use of fourth generation techniques

(Section 2.10). Rather than creating software using conventional third generation

programming languages the RAD process works to reuse existing program components

(when possible) or create reusable components (when necessary). In all cases,

automated tools are used to facilitate construction of the software.

**Testing and turnover.** Since the RAD process emphasizes reuse, many of the program

components have already been tested. This reduces overall testing time. However,

new components must be tested and all interfaces must be fully exercised.

33

Business

modeling

Team #1

Data

modeling

Process

modeling

Application

generation

Testing

&

turnover

Business

modeling

Data

modeling

Process

modeling

Application

generation

Testing

&

turnover

Data

modeling

Process

modeling

Application

generation

Testing

&

turnover

Team #2

Team #3

60–90 days

Business

modeling

**FIGURE 2.6**

The RAD

model

34 PART ONE THE PRODUCT AND THE PROCESS

The RAD process model is illustrated in Figure 2.6. Obviously, the time constraints

imposed on a RAD project demand “scalable scope” [KER94]. If a business application

can be modularized in a way that enables each major function to be completed

in less than three months (using the approach described previously), it is a candidate

for RAD. Each major function can be addressed by a separate RAD team and then

integrated to form a whole.

Like all process models, the RAD approach has drawbacks [BUT94]:

• For large but scalable projects, RAD requires sufficient human resources to

create the right number of RAD teams.

• RAD requires developers and customers who are committed to the rapid-fire

activities necessary to get a system complete in a much abbreviated time

frame. If commitment is lacking from either constituency, RAD projects will

fail.

• Not all types of applications are appropriate for RAD. If a system cannot be

properly modularized, building the components necessary for RAD will be

problematic. If high performance is an issue and performance is to be

achieved through tuning the interfaces to system components, the RAD

approach may not work.

• RAD is not appropriate when technical risks are high. This occurs when a new

application makes heavy use of new technology or when the new software

requires a high degree of interoperability with existing computer programs.

# **What is Iterative Model in SDLC?**

[January 26, 2017](http://hackingig.com/what-is-iterative-model-in-sdlc/) [admin](http://hackingig.com/author/admin/) [0 Comments](http://hackingig.com/what-is-iterative-model-in-sdlc/#respond) [SDLC](http://hackingig.com/tag/sdlc/), [Software testing](http://hackingig.com/tag/software-testing/)

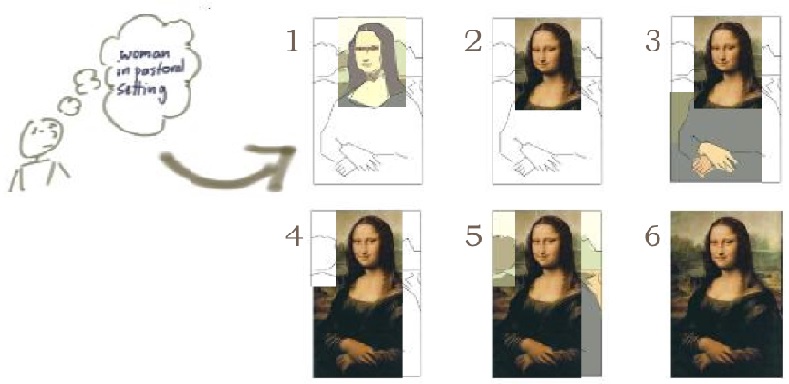
Iterative development is a way of splitting the development of a large application into smaller blocks. In iterative life cycle model the development does not start with a full specification of requirements, instead the development begins by featured requirements.

The developed model can be reviewed to identify future requirements. This process is repeated producing new builds of the software for each cycle of the model.In iterative model we cycle through a number of smaller self-contained life cycle phases for same project or software. Each iteration adds some new functionality to existing build.

The increment has the infrastructure required to support the initial build functionality. The build which is produced by an iteration may be tested at several levels as part of its development.Subsequent increments will undergo testing for the added functionality, regression testing of existing functionality and integration testing of both added and existing functionality.

Regression testing is of most importance on all subsequent builds after the first one. In iterative development process more testing effort is required at each subsequent delivery phase which is considered in project plan. As the build is produced in iteration pattern, software or product can come in market early with critical functionality and validation testing is carried out at each increment, thereby giving early feedback on fitness-for-use of product and its business value.

It is simpler to manage because the complete workload is divided into small small pieces. In development using iteration model initial investment cost is less although it may cost more in long term.

[](http://hackingig.com/wp-content/uploads/2017/01/Iterative-model-SDLC.jpg)

In the above diagram painter has worked iteratively on his painting of monalisa. In the same way in iteratively we create rough product or product piece in one iteration, then review it and improve it in next build.

Step 1: Developer start the project but quickly realize, it wasn’t a good idea and give up completely.They lost the money it took to develop it but avoided paying the full cost of the project. Time to stop.

Step 2: Developers built the top priority functionality (Mona Lisa’s head) and they launch with that. At this point they are not sure do they need to build the rest as the return on investment will be too low. Time to stop.

Step 3: In this iteration developers build out the second tier of priorities (Mona Lisa’s hands and body). This build is not fully featured but filled in a few gaps that significant customers complained about but they realized the rest of the scope isn’t really needed. Time to stop.

Step 4: In this iteration developers have built all the must have and should have items from the original high level requirements list (Mona Lisa’s head, hands and body). There are other things they could have given but the timeline is over or the business is starting up a new product line and needs to divert development effort to that. Time to stop.

Step 5: They got all the major features and, in fact, we’ve touched on all the features from the product backlog. There is more they could do but really that would be more like gold plating than delivering essential functionality. Time to stop.

Step 6: Wow superb and impossible to improve further. Time to stop.

**Advantages of Iterative Model:-**

1. More time is given to designing the model and less effort is spent on documentation.

2. As builds are released in iteration we can reliable user feedback.

3. Defects are tracked in early stages are builds are released in iteration.

**Disadvantage of Iterative Model:-**

1. As development of built starts without gathering complete requirement design issue may arise in future.

2. Costly system architecture

**When to use Iterative model:-**

When the project is big.

PROJECT PLANNING OBJECTIVES

The objective of software project planning is to provide a framework that enables the manager to make reasonable estimates of resources, cost, and schedule. These estimates are made within a limited time frame at the beginning of a software project and should be updated regularly as the project progresses. In addition, estimates should attempt to define best case and worst case scenarios so that project outcomes can be bounded.

The planning objective is achieved through a process of information discovery that leads to reasonable estimates.

What is it? Risk analysis and

management are a series of steps

that help a software team to

understand and manage uncertainty. Many problems

can plague a software project. A risk is a

potential problem—it might happen, it might not.

But, regardless of the outcome, it’s a really good

idea to identify it, assess its probability of occurrence,

estimate its impact, and establish a contingency

plan should the problem actually occur.

Who does it? Everyone involved in the software

process—managers, software engineers, and customers—

participate in risk analysis and management.

Why is it important? Think about the Boy Scout motto:

“Be prepared.” Software is a difficult undertaking.

Lots of things can go wrong, and frankly, many

often do. It’s for this reason that being prepared—

understanding the risks and taking proactive measures

to avoid or manage them—is a key element

of good software project management.

What are the steps? Recognizing what can go

wrong is the first step, called “risk identification.”

Next, each risk is analyzed to determine the likelihood

that it will occur and the damage that it

will do if it does occur. Once this information is

established, risks are ranked, by probability and

impact. Finally, a plan is developed to manage

those risks with high probability and high

impact.

What is the work product? A risk mitigation, monitoring,

and management (RMMM) plan or a set of risk information sheets is

produced.

How do I ensure that I’ve done

it right? The risks that are analyzed and managed

should be derived from thorough study of

the people, the product, the process, and the project.

The RMMM should be revisited as the project

proceeds to ensure that risks are kept up to

date. Contingency

QUICK

LOOK

146 PART TWO MANAGING SOFTWARE PROJECTS

Peter Drucker [DRU75] once said, "While it is futile to try to eliminate risk, and

questionable to try to minimize it, it is essential that the risks taken be the right risks."

Before we can identify the "right risks" to be taken during a software project, it is

important to identify all risks that are obvious to both managers and practitioners

RISK IDENTIFICATION

*Risk identification* is a systematic attempt to specify threats to the project plan (estimates, schedule, resource loading, etc.). By identifying known and predictable risks, the project manager takes a first step toward avoiding them when possible and controlling them when necessary.

RISK PROJECTION

*Risk projection,* also called *risk estimation,* attempts to rate each risk in two ways—the likelihood or probability that the risk is real and the consequences of the problems associated with the risk, should it occur. The project planner, along with other managers and technical staff, performs four risk projection activities: (1) establish a scale that reflects the perceived likelihood of a risk, (2) delineate the consequences of the risk, (3) estimate the impact of the risk on the project and the product, and (4) note the overall accuracy of the risk projection so that there will be no misunderstandings.

RISK REFINEMENT

During early stages of project planning, a risk may be stated quite generally. As time passes and more is learned about the project and the risk, it may be possible to refine the risk into a set of more detailed risks, each somewhat easier to mitigate, monitor, and manage.

RISK MITIGATION, MONITORING, AND MANAGEMENT

All of the risk analysis activities presented to this point have a single goal—to assist the project team in developing a strategy for dealing with risk. An effective strategy must consider three issues:

• risk avoidance

• risk monitoring

• risk management and contingency planning

If a software team adopts a proactive approach to risk, avoidance is always the best strategy.

*Risk management and contingency planning* assumes that mitigation efforts have failed and that the risk has become a reality. Continuing the example, the project is

“We are ready for an unforseen event that may or may not occur.”

well underway and a number of people announce that they will be leaving. If the mitigation strategy has been followed, backup is available, information is documented, and knowledge has been dispersed across the team. In addition, the project manager may temporarily refocus resources (and readjust the project schedule) to those functions that are fully staffed, enabling newcomers who must be added to the team to “get up to speed.” Those individuals who are leaving are asked to stop all work and spend their last weeks in “knowledge transfer mode.” This might include video-based knowledge capture, the development of “commentary documents,” and/or meeting with other team members who will remain on the project