Prototyping

A prototype is:

- *an original or model after which anything is formed;*
- *the first thing or being of its kind;*
- *a pattern, exemplar, or archetype* " (Websters 20th C.)

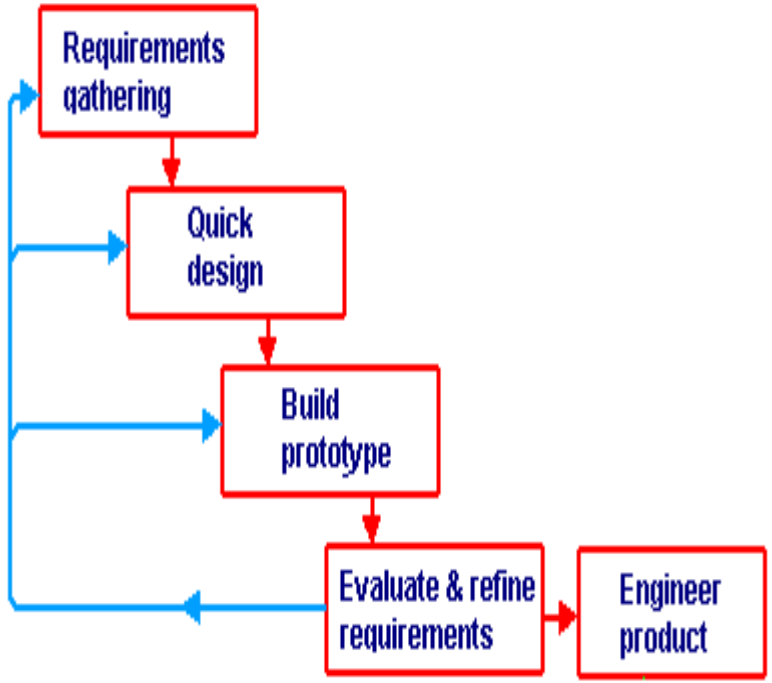
Prototyping in systems development is the process of creating a 'cut-down' version, or part, of a system so that users can:

- get an idea of what the system will offer; and
- provide feedback on whether the system is what is required.

Prototyping helps to identify misunderstandings between 'users' and software developers; and may detect missing (ie: not yet specified) user requirements.

Advantages of prototyping stage:

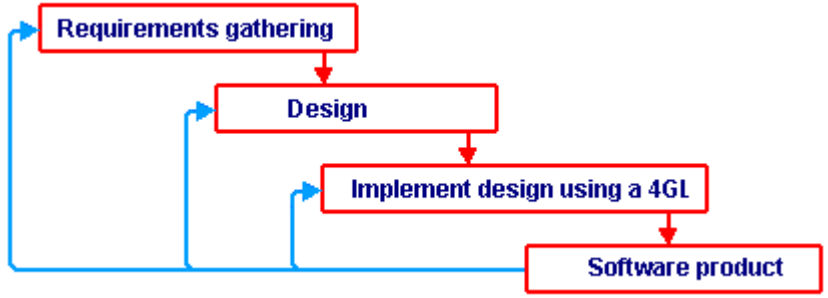
- identification of misunderstandings between 'users' and software developers;
- detection of missing user services
- identification of difficult-to-use or confusing user services
- requirements validation aid (discovering inconsistencies)
- early availability of a working (limited) system
- may serve as basis for specification for a 'production quality' system



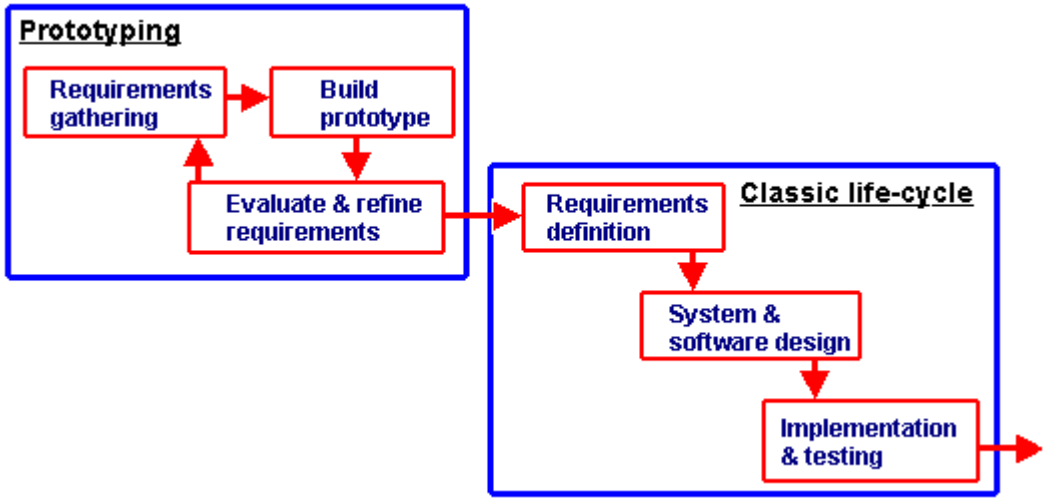
4GLS Currently, 4GLs are good tools for prototyping. The 4GL process model is almost identical to the prototyping model.

Software development using a 4GL

Prototyping with 4GLs can be very useful in the requirements analysis and definition stages of large projects which otherwise 'use' a traditional lifecycle process model.



Prototyping as analysis tool



Advantages of using a 4GL in prototyping stage

- improved productivity in software engineering and (possible) reduction in costs;
- complete rewrites of prototype software possible allowing (possibly) a more exploratory approach; and also
- (possibly): 'user' programming;
- improved documentation;
- smaller development teams.

Software Development Methodology

Choosing the right technologies and tools are vital for the success of any software development process. The right decision here could ensure the project's ultimate success. In this article, we try to look into what is the right approach when starting a new project.

In the last few decades, we have seen the software world transform immensely. We learned to be transparent and agile, to be lean, to approach our users and clients for feedback more often. We learned to analyze and use data before designing a product. It changed the nature of the software business forever, and companies such as Google, Microsoft and many others innovated new frameworks not only for writing code but to manage the projects lifecycle as well.

What to consider when choosing the right software development methodology

There is a fundamental difference between processes within a product and for example, outsourcing companies and how they handle the clients' requirements. Choosing a software development methodology simply because it is trendy does not mean that you would manage to implement it properly and benefit from it, quite the contrary. If we select the wrong methodology and implement it incorrectly, it can jeopardize the whole project.

To make the right choice, we need to look at the structure of the team and the type of project that we plan to manage:

– Dedicated team

– Fixed-price model

– Support and Maintenance

Each of them comes with different challenges. By choosing the right project management framework, we can successfully tackle these challenges in each case.

How to Choose the Best Software Development Methodology

Choosing the right methodology is sometimes overwhelming. The following tips may help companies choose.

1. Requirements Flexibility – Before deciding on a methodology, teams should consider the requirements of the project.
2. Project Size – Some methodologies work best for large-scale projects, while others are more suitable for smaller projects. Teams should evaluate the size of the project as a factor in deciding the right methodology.
3. Time Required – Some methodologies help reduce the time to market for projects. If the project is on a strict time schedule, that’s definitely something to keep in mind while deciding.

How to Choose the Right Software Development Tool?

Selecting a technically sound developer software tool for a project can make or break the deal. Every project has unique needs that need to be met through brainstorming the exact fit for your project. Consider the following points to go about this:

1. Project essentials

Know your project inside-out to nail this part. Every project has needs so specific that there is only one right software developer tool with the features you're looking for. This will enable you to choose the perfect fit equipped with technologies available at your disposal. In the end, you will look at a system ready to serve the best results for the resources and time. How it plays a role in the final execution of the project is a crucial question to answer.

2. Team experience

The team's experience, knowledge, and comfort with the shortlisted web developer software tools need to be considered. Discussions among the developers based on their hard-earned knowledge will help get an idea of everyone's opinions. Make sure all your team members are heard before you settle on a choice.

3. Feedback

Whether or not the project's expectations meet reality can be determined through feedback. While you may not think much of it in the present moment, this can be a useful reference to turn back to in hindsight. It is a key factor in not repeating the same mistakes in future trials.

4. Project configuration

Give in-depth thought to the tentative scope of your project. It will aid you in narrowing down the options of software developer productivity tools. Some software is built for small-scale projects. To avoid running into this issue after all the set-up is done, think it through for your project to be on the same page as the available testing cycles.

Best Software Development Tools You Should Know

#1) UltraEdit
#2) Quixy
#3) Embold
#4) Jira
#5) Linx
#6) GeneXus
#7) Zoho Creator
#8) Delphi
#9) Atom
#10) Cloud 9
#11) GitHub
#12) NetBeans
#13) Bootstrap
#14) Node.js
#15) Bitbucket
#16) CodeCharge Studio
#17) CodeLobster
#18) Codenvy
#19) AngularJS
#20) Eclipse
#21) Dreamweaver
#22) Crimson Editor
#23) Zend Studio
#24) CloudForge
#25) Azure
#26) Spiralogics Application Architecture (SAA)

Fourth Generation (Programming) Language (4GL)

A fourth generation (programming) language (4GL) is a grouping of programming languages that attempt to get closer than 3GLs to human language, form of thinking and conceptualization.

4GLs are designed to reduce the overall time, effort and cost of software development. The main domains and families of 4GLs are: database queries, report generators, data manipulation, analysis and reporting, screen painters and generators, GUI creators, mathematical optimization, web development and general purpose languages.

Also known as a 4th generation language, a domain specific language, or a high productivity language.

Fifth Generation (Programming) Language (5GL)

A fifth generation (programming) language (5GL) is a grouping of programming languages build on the premise that a problem can be solved, and an application built to solve it, by providing constraints to the program (constraint-based programming), rather than specifying algorithmically how the problem is to be solved (imperative programming).

In essence, the programming language is used to denote the properties, or logic, of a solution, rather than how it is reached. Most constraint-based and logic programming languages are 5GLs. A common misconception about 5GLs pertains to the practice of some 4GL vendors to denote their products as 5GLs, when in essence the products are evolved and enhanced 4GL tools.

Also known as a 5th generation language.

End-User Computing (EUC)

End-user computing (EUC) refers to computer systems and platforms that are meant to allow non-programmers to create working computer applications. It is a compilation of approaches meant to better involve and integrate end users and other non-programmers into the world of computing systems development. EUC is broad and may encompass different meanings that are more or less related, but has the overarching context of allowing end users to better control their computing environment without the aid of real programmers or developers, such as an accountant using Microsoft Excel to automate his/her tasks.

Fourth Generation (Programming) Language (4GL)

This is a high-level programming language, which is used by database users to access the database.
This is also called non-procedural language because just like other programming languages it does not follow any fixed procedure or sequence for execution. Instead, it allows users to just pass on the commands in simple English text that follow simple syntax which can be easily understood by any user like this: create table, select data, insert data, etc.
SQL, Informix 4GL, and Oracle are examples of 4gl.

Advantages of 4GL

- Now days databases are used everywhere to manage data so 4GL makes it very easy to create, manage and operate the databases.
- A single line command can perform the task. On the other hand, in other languages, we need to write a series of commands (sometimes a huge segment) for the same task, in which syntax and keywords are not easily understandable.
- This type of language just focuses on “what is required”.
- Users need not worry and define “how it needs to be performed”.
- It is very easy and simple to use even for beginners or end users.
- It reduces overall cost, time, and effort.

Disadvantages of 4GL

- This language is only database oriented, which means we can use it for databases only.
- This is easy for users but in backend each query executes a sequence of commands which makes it time consuming. (Not create that much delay and effect)

Main Components of 4GL

- Database Language and Queries
- Report Generators
- Analysis and reporting
- GUI creators
- Mathematical optimization
- Spreadsheets
- To create an interface application for end users

Different Stages of System Investigation

Preliminary investigation is the first step in the system development project. It is a way of handling the user's request to change, improve or enhance an existing system. System investigation includes the following two stages:

1. Problem definition
2. Feasibility study

1. Problem definition:

The first responsibility of a system analyst is to prepare a written statement of the objectives of the problem. Based on interviews with the user, the analyst writes a brief description of his/her understanding of the problem and reviews it with both the groups. People respond to written statements. They ask for clarifications and they correct obvious errors or misunderstandings. That is why a clear statement of objectives is important. In other words, proper understanding of the problem is essential to discover the cause of the problem and to plan a directed investigation by asking questions like what is being done. Why? Is there an underlying reason different from the one the user identifies? Following are some possible definitions of problems:

a. The existing system has a poor response time
b. It is unable to handle the workload.
c. The problem of cost, that is the economic system is not feasible.
d. The problem of accuracy and reliability
e. The required information is not produced by the existing system
f. The problem of security.

2. Feasibility study:

The actual meaning of feasibility is viability. This study is undertaken to know the likelihood of the system being useful to the organization. The aim of feasibility study is to assess alternative systems and to propose the most feasible and desirable system for development.

Thus, feasibility study provides an overview of the problem and acts as an important checkpoint that should be completed before committing more resources. The feasibility of a proposed system can be assessed in terms of four major categories as given below:

a) Organizational feasibility: the extent to which a proposed information system supports the objective of the organization's strategic plan for information systems determines the organizational feasibility of the system project.

b) Economic feasibility: In this study, costs and returns are evaluated to know whether returns justify the investment in the system project.

c) Technical feasibility: whether reliable hardware and software, capable of meeting the needs of the proposed system can be acquired or developed by the organizations in the required time is a major concern of the technical feasibility.

d) Operational feasibility: the willingness and ability of the management, employees, customers, suppliers, etc to operate, use and support a proposed system come under operational feasibility. In other words, the test of operational feasibility asks if the system will work when it is developed and installed.