UNIT 1 WEB SOFTWARE ENGINEERING

Structure **Page Nos.**

1.0 INTRODUCTION

Today's Internet age has brought about a large demand for services that can be delivered over the network. These services rely in large measure on web software applications that can be accessed concurrently by many users. Increasing competition has meant that vendors want shorter and shorter cycle times for producing new products in any sphere, but in the arena of web based services, the situation is extreme. Instead of years or even months, new applications have to be readied and deployed in weeks, before someone else does so. Such pressures are compounded by the need to have high quality software as no one will visit a buggy website for the second time.

Besides technical challenges of architecture and design, web software brings in difficulties caused by the peculiarly distributed nature of the whole endeavour. Not only is the architecture such that the different tiers could be working out of potentially different machines located widely apart, the users are likely to be geographically separated by vast distances and could come from different cultural backgrounds. And, as likely as not, the development team could be spread out in space and these greatly separated team members being responsible for the software produced.

Web software could be –

- Meant for the Internet, with users coming from the public in a geographical region that could perhaps be the whole world.
- Meant for a closed group such as a corporation, in which case, it would be running over an Intranet.
- Meant for an extended, but still closed groups such as corporations, its customers and suppliers, where upon it is an Extranet.

Irrespective of the type of usage intended, the users could be separated by large distances and the application needs to behave correctly for all of them. In this unit, we will concentrate on web applications meant for the Internet.

We will also explore the difficulties associated with developing and managing a web based software project. Also examined will be some approaches and general rules

that could help us to do such a project well. Please be aware that the subject is vast, and in this short unit we can only touch upon the complexities of the topic.

1.1 OBJECTIVES

After going through this unit, you should be able to:

- identify the characteristics of a web based software application;
- describe the architecture of such an application;
- identify the challenges of managing a web based software project;
- think up relevant metrics to gauge the health of such a project;
- ensure proper understanding of the requirements in such a project so that development can be pursued, and
- facilitate quality control of a web based project.

1.2 DIFFERENT CHARACTERISTICS

While web based software applications can vary a lot in size and complexity, all of them share common characteristics, some of which are peculiar to them and are not prominent in other applications. We can look at these characteristics in terms of–

- The way the applications are developed, tested and deployed
- The way the applications are used
- The way they are maintained
- The design of these applications.

1.2.1 Characteristics of a Web Application

In this section, we will look at the overall architecture and deployment context of web based applications and see how this leads to their peculiar management challenges. Unlike conventional applications that can be monolithic, web applications are by their very nature amenable to layering. One cannot have a monolithic web application as the client and the rest of the application have to be of necessity separated. In principle, one could have a thick client application with a lot of intelligence embedded at the client end, but that would defeat the very purpose of delivering an application over the web. The idea is to have a client that is common to all users so that anybody can access the application without having to do anything special. This client is typically a browser of some sort that can render a HTML page on the user's screen.

Most of the processing is done at the other end of the application, that is, at the server. Here again there can be separation between the application and the data storage in the database. These two layers can be on separate machines that are themselves separated over the network.

The layering approach can bring about much greater flexibility and simplicity in design, maintenance and usage. In a monolithic application, everything is coupled together and in a layered application, we can change one layer without affecting the behaviour of others. For example, the business logic at the application layer can be changed without affecting the user interface or the database design. We can change the database management system from one vendor's offering to another without changing the application code or the user interface.

There can be many kinds of web applications such as –

- Those with static text
- Content that is changed frequently
- Interactive websites that act on user input

- Portals that are merely gateways to different kinds of websites
- Commercial sites that allow transactions
- Those that allow searches.

1.2.2 Development, Testing and Deployment

Although many basic processes and methodologies do not change, there are some features of the development cycle of web applications that are different. While some of these will be looked at in more detail later in the unit, we need to understand these features to be able to effectively develop a web application project.

Development

The development of such an application has to take into account the fact that there is now one more element in the whole behaviour of the application – the network. The element of communication could be ignored in conventional software, but it plays a pivotal role in web based projects. Different users can be using networks that are slow or fast and everything in between. Networks are not fully reliable and communication lines do drop or fluctuate in their speed. While network protocols will take care of the issues of data reliability, this comes at the cost of uniform speed. Designers of web applications need to take care of situations like race conditions, lost connections, timeouts and all the other tiresome realities of wide area networks.

Web applications, if used over the Internet, can potentially have a very large number of users. So we need to be careful about matters like machine configurations and sizing. It is also frequently necessary to have a scalable architecture that can be upgraded as application loads rise with the growing popularity of the site. And if the site becomes really popular, this need can be very frequent indeed!

Testing

The testing of web applications has its own difficulties, caused mainly by the variable speed with which inputs reach the server and responses reach the user over the World Wide Web. It is hard to simulate real Internet like conditions in test environments that are typically based on a local area network.

Testing tools do exist that allow inputs to be sent from machines located in different parts of the world to the application. While comparatively expensive, they do allow for realistic testing of the applications. There are other ways of testing the applications by trying to simulate delays and other conditions of the real Internet. To be able to obtain best, average and worst case response times, it is necessary to run tests many times. Automated testing is important for such load, performance and stress tests. They cannot be realistically performed manually.

Although interoperability and consistency are assumed to be the good features of standardized browsers and databases, in practice there are several problems that can arise because of the differing behaviour of browsers caused by–

- Different operating systems such as Linux, Windows versions or Unix systems.
- Different versions of the operating systems.
- Differences in other components such as the Java Virtual Machine.
- Differences in behaviour of the browsers themselves because of extensions or incomplete adherence to standards.
- Bugs in certain versions of the browsers on certain operating systems.

It is often impractical to test the application on all possible combinations of browsers, operating systems, databases and their various versions. In such cases the combinations for which tests have been performed are sometimes indicated and the

appropriate fixes have to be made to the application as bugs are discovered and reported during field use.

Deployment

In most conventional application rollouts, there is often a fair degree of control available to the developers because of the possibility of limiting access. Limited releases to a select test group can be made so that the most obvious and common errors and annoyances are taken care of before the regular release. But in a web application that works over the Internet, it can be difficult to do such tests. True, access can be limited by not publicising the URL at which the application is available, or by having access control in some form that is known only to the select band of users in the test group. Still, since the application is to be available publicly, there is a conflict between the small group which gets early access and the possibility of realistic testing. The very characteristic of a web application is having a large body of concurrent users.

We have already mentioned the need for scalability in the hardware. This is particularly important in terms of storage. We need to be careful that we do not quickly run out of space, for example, in an e-mail or storage site. Besides, it might be important to have the capability to beef up the bandwidth available to the server that is hosting the application, lest it get swamped by user requests.

1.2.3 Usage of Web Applications

Applications hosted on the web are meant to be used by lay users who may have little knowledge of computers. Conventional applications also have to deal with usage by non- technical people, but there we can often organise training or help desks to take care of any problems that may arise. That possibility is not available for a web based application. Because, we cannot reach and train all of our target audience, even if it were possible to identify them individually. As any developer with a little experience knows, users can "exercise" software in ways that were never anticipated.

Normally, if a user comes across difficulties while using the application, she could be expected to report the problem to the appropriate group in the organisation or to the vendor responsible for maintaining and deploying the software. But in the case of an application on the web, the user may simply give up and just not come back to the site if she is not able to work the page, and that too in the first few seconds. Or, her network connection might break while trying to use the site, and that could well be her last visit. So, the tolerance level of users is exceptionally low in such applications.

Users can also be expected to want a fast response and if some operation is likely to take more than a few seconds, it will be best to keep them informed. Feedback on the state of a request is also necessary, as also instructions on what to do if they are disconnected in the middle of a long operation.

An application on the web is much more likely to be internationalised as it might have to cater to the needs of users from different parts of the world. You may be shutting out a large number of potential visitors to your application if you restrict yourself to any particular language. You should also expect to see little data entry and as much as possible of features such as auto completion, drop down menus, look up tables and the like.

1.2.4 Maintaining Web Applications

The maintenance of web applications has its own set of challenges. Frequent changes to the presentation, or look and feel are expected. Many of the static text pages may need updating ever so often and some of the content could be changing daily or hourly. So the task of maintenance has to expand to take care of changing content and this can be a major exercise in some sites. Business rules could also be volatile, with new schemes and packages being frequently invented to entice the users. It means that the maintainers have to take care of frequently changing the business logic with little change to the other layers.

People are sensitive about sites that are perceived as not trendy enough and expectations are high as regards features, good looks and easily usable interfaces. One might have to work hard at changing and upgrading these to keep up with the latest advances. If a site is not changed for more than a few months, it will not be looked upon as live or up to date. So, taking care of a web application is often much more than just fixing bugs or updating the software to implement changes to business rules. There is a fair amount of effort that has to be put in to improve usability.

1.2.5 Designing Web Applications

Since web applications are layered, we get a simplified design that can be easily maintained and altered. However, that also produces the corresponding need to design the different components to be generalised enough so that they can be useful to a larger audience. Applications could be running under the control of an application server for which the designers have to create and implement the requisite beans. For some purposes, beans can be purchased from other software vendors or they could have to be written from scratch for the application.

These days, some functionality in the application could be coming out of web services, components on the web that can run with your application and that provide defined behavior. An application could look for an available web service that can do some needed action. Web services advertise their available functionality that needy applications can tap. This distributed nature of the application greatly increases the amount of reusability, but brings about problems of security and reliability.

The user interface in a web application is very important indeed and can often be the tipping point in the success of a website. Ease of use and ergonomic considerations are important. The application has to be able to cater to users of disparate skill levels and in some countries may be required to provide access to people with disabilities.

All the above characteristics need to be kept in mind when planning and otherwise managing a web based application project.

Check Your Progress 1

1) The _____ can bring about much greater flexibility and simplicity in design, maintenance and usage of web applications.

2) One of the significant characteristics of a Web Application is to have a large number of _____ users.

1.3 ISSUES OF MANAGEMENT OF WEB BASED PROJECTS

Like any other software application project, we need to use good software development practices when faced with working on a web application. Otherwise the project would not remain in control and we would face problems with timeliness, budgets and quality. Before going on to the special issues, it would be useful to review the general good practices that we need to follow.

1.3.1 Review of Good Software Development Practices

There is by now a wealth of literature and experience on how to manage software projects. While two decades ago the software crisis was a big stumbling block and failed or overdue projects were commonplace, the efforts of quality gurus have resulted in much better techniques of management that address these problems. Many projects still have trouble, but that is not because the techniques were not known but because they were not followed.

There are several quality models available to organisations that can be followed, such as the ISO 9001:2000 quality system model or the Capability Maturity Model Integration (CMMI) of the Software Engineering Institute of the Carnegie Mellon

University, Pittsburgh, USA. Let us look now at some of the basic, common principles of software management practice.

Managing Requirements

It is very important in any project to have a clear understanding of the requirements shared with the customer. Very often the developers have no notion of how the software will be used or what would be nice for the user to have. While a requirements document might be prepared and approved by the customer, there are many issues that are hard to capture in any document, despite having detailed discussion with customer. To ensure that the viewpoints of the developers and customer do not diverge, regular communication is a must. Finer points that could not be captured in the documentation have to be understood by the project team through proper coordination with the customer. Here the customer could be an external party or could be another group within the same organisation, in which case it is an internal customer.

Even if we achieve a good understanding of the requirements, software projects are notorious for changes. This means that we must have a good system of tracking requirements as they change. The rest of the artifacts in the project must reflect the changes that have occurred in the requirements.

Managing the Project

The essence of managing the project is proper planning for the project and then executing the project according to the plan. If the project deviates from the plan, we need to take corrective action or change the plan. We must always be proactive and keep looking for the risks in the project, rather than react to problems after they have arisen. We should try to anticipate what can go wrong and take action accordingly. Here a plan needs to keep something apart from the schedule of the project.

The project plan is the basic document used to execute the project and has all the required information about it. It should be the guiding document for the project and hence must be kept up to date. Some of the items of information in the plan could be–

- Background information and context
- Customer and other stakeholder information
- Estimates of cost, effort and duration
- Dependencies on outside groups
- Resource requirements – equipment and people
- Methodology to be used to do the project
- How the project will be monitored.
- Schedule.

Besides the project management plan, there are various subordinate plans that need to be made for different aspects of the project, such as,

- Configuration Management Plan
- Quality Assurance Plan that covers audits and process compliance
- Quality Control Plan that covers technical reviews and testing
- Risk Management Plan
- Training Plan
- Metrics Plan
- Defect Prevention Plan

There should be regular communication between the team members themselves to ensure that there is a shared vision and sense of purpose, and that perceptions remain aligned to the project goals.

As the project is being executed, there has to be regular monitoring of the progress and of the different parameters of the project, such as effort, cost and quality. If any of these parameters is not in line with what was planned, appropriate investigation, analysis and corrective action needs to be taken. Sometimes circumstances change such that it is no longer possible to bring back the project on track with the original plan, whereupon the plan needs to be reviewed and revised.

Configuration Management

This is vital to retaining control of the artifacts produced by the project. Given that changes will occur, we need to be able to track and know at all times which version of an artifact is current and which ones are obsolete. We do not discard obsolete versions because it may sometimes be necessary to backtrack or to branch off in another direction from a common base.

Whenever an artifact is produced it should be reviewed so that we can have confidence in the veracity and appropriateness of its content. After this it should be placed in a baseline area and any further changes to it should be done only according to formal change procedures. We have to plan for which artifacts will be so controlled.

Access to baselined artifacts should be limited to a designated configuration controller. Any member of the team who needs to alter a baselined artifact needs to approach the configuration controller who will then check out the artifact and allow the team member to make the change. Simultaneous checking out by more than one team member is discouraged, but if done, any conflicting changes that may be made by them need to be taken care of.

Measurements

It is important to measure our software work so that we can keep control of the proceedings. This is the key to being able to manage the project. Without measurements we will not be able to objectively assess the state of the progress and other parameters of the project, and we would be reduced to relying on our intuition and feel for where the project stands.

Software engineers have been notorious for not bothering to measure their work, or for insisting that their work cannot be measured. It is an engineering discipline unique in that measurements have not been given. However, in the 1990s, great progress was made in spreading awareness and appreciation of the need for software measurements. Still, even today, many organisations do not have a good metrics program going.

Risk Management

An important constituent of project management is managing the risks of the project. A risk is an event that can have a deleterious impact on the project, but which may or may not occur. Thinking of the possible risks is the first step in trying to take care of them. While we cannot get rid of risks, we need to keep thinking of ways to reduce their effects. Better still, we should try to take steps that will prevent the risk from becoming a reality.

Not all risks need to be looked at. When we analyse risks, there are some that are more probable and also have a significant adverse effect on the project. Others may have a severe effect if they occur, but are not too likely to happen. Still others may not be too harmful, even if they do come to pass. There can be a large number of risks in a project, but thinking about all of them is not practical. We need to concentrate on the most important risks and direct our energies to managing them.

Over time, the risk parameters can change. We, therefore need to keep re-looking at our risks and alter our risk plan to take cognizance of the changed situation as the project progresses. We should keep carrying out the actions that make the risk less likely. Whenever a risk does occur, we should put into action our plan for reducing its harmful effects.

Thus risk management allows us to look at a project actively, rather than passively reacting to events after they have occurred.

1.3.2 Organisation of Web Application Teams

Web applications tend to need much more ongoing support, maintenance and enhancement than others. After the initial application has been rolled out, comes the stage of maintenance. This can amount to weekly or even daily releases in the first instance while the software stabilises. This endeavour requires a team structure that is a bit different from others. We start from the maintenance end to emphasise its importance, and the fact that design and development are done comparatively quickly. Sometimes the need for project management during development itself is questioned but, as in any other project, good management is critical to success in a web application project as well.

Webmaster

This role is not unique to web applications, but is usually otherwise referred to as the administrator. It entails taking care of the site on a day to day basis and keeping it in good health. It involves close interaction with the support team to ensure that problems are resolved and the appropriate changes made. Some of her activities include –

- Gathering user feedback, both on bugs (such as broken links) and also on suggestions and questions from the public. These need to be passed on to the support team who are engaged in adapting the site in tune with this information.
- Ensuring proper access control and security for the site. This could include authentication, taking care of the machines that house the server and so on.
- Obtaining statistics and other usage metrics on the site. This could include, among many others –
    - Number of hits, distributed by IP address
    - Number of registered visitors
    - Frequency distribution of the different pages visited
    - Bandwidth utilised for upload and download
    - Number and type of errors, particularly of service denials.
- Helping to ensure that change control procedures are followed.
- Archiving old content while keeping newer content in a more prominent and easy to find location.

Application Support Team

In conventional software projects, while maintenance is important, it may or may not be done by the organisation that developed the project. In web applications, the organisation that developed the site is quite likely to be given the responsibility of its maintenance. This is because web applications tend to keep evolving and what corresponds to the development phase in a conventional application is here quite brief and frenetic. A large part of the evolution of the software tends to happen subsequently, based on user feedback and continuing refinement of the concept by those who conceived of the application. The activities here can consist of–

- Removing bugs and taking care of cosmetic irritants.
- Changing the user interface from time to time for novelty, to accommodate user feedback and to make it more contemporary.
- Altering the business rules of the application as required.
- Introducing new features and schemes, while disabling or removing older ones.

The support team could consist of designers including graphic artists, programmers, database specialists and testers.

Content Development Team

In conventional business application software, there are usually changes and modifications required to master tables in databases, such as altering price lists, available items and so forth. Web applications frequently require much more ongoing, sustained effort to retain user interest because of the need to change the content in the site. In the case of a news site, this updation can be every few minutes. The actual news could be coming from a wire feed from some news agencies, or from the organisation's own sources.

Though not all sites need to be so current, it is often required to keep adding articles and papers of interest to the user community. These stories or articles are written by a team that has expertise in the domain concerned. A site could be serving out a wide variety of information and other stories from different interest areas. Content development teams could be researchers who seek out useful or interesting facts, authors, experts in different areas, and so on. The content may be edited before being served out.

The members of this team are usually not knowledgeable from the software point of view, though they might be well respected in their own domains. They may be full time members of the organisation or casual, freelance contributors. So though called a team, some of them might really be individual providers. The content they produce is typically in the form of a word processor file that might have images, graphs, tables and other non-text ingredients.

Other forms of content are now quite commonplace, such as audio files, slide presentations or video clippings of events, places or speakers. With increasing availability of high speed network connections for even individual users, such content is now becoming increasingly popular because of the impact it can produce.

Web Publisher

This is an important role that connects the content development team to the actual website. The raw material created by the writers has to be converted into a form that is suitable for serving out of a webserver. It means formatting the content according to the requirements of a markup language such as HTML. Though tools can help perform much of this, there still might be human effort that is needed.

In the case of automated news feeds, such publishing needs to be as tool driven as possible. The web publisher must have a good understanding of the technical aspects of web servers and the markup language.

1.3.3 Development and Maintenance Issues

Let us now look at some of the management issues that we will face while developing and maintaining a web application. We have already seen that web applications tend to evolve, and there is little distinction between what in a more conventional project we would refer to as development or maintenance. It is hard to say when the development is complete and maintenance starts in a web project. In practice it is a continual cycle of delivering something and then of fixing and correcting and enhancing it.

Configuration Management

During maintenance of a web application, because of the frequency of changes, configuration management assumes considerable importance. The change control process must be up to the demands placed on it of keeping control over the software configuration inspite of the many changes that will happen. So it must not be overly elaborate or hard to follow. That would mean it might not get followed and the consequences will be disturbing. Once one loses track of what is happening in the software it can become very difficult to get back. It will then become impossible to

predict the effect of changes to the software and one would have to grapple with mysterious bugs whose cause will be hard to find.

Besides the software itself, content also needs to be subjected to configuration management. This could include items such as information on schemes or other service or product offerings. Lack of discipline here could mean horrors like –

- Postings on past offerings that are no longer available,
- Incorrect prices or other terms being visible to users,
- Postings of content that is still work in progress and was not ready for publication apart from lesser embarrassments such as archival content reappearing on the site.

A very important aspect is that of inadequately tested code finding its way into the active configuration. It could mean that visitors to the site have to put up with features that do not work as expected or simply crash on them. Worse, it can result in security holes that compromise the security and integrity of the site. Sometimes, the privacy of data of registered users could be in question. In other cases it could expose other systems of the organisation itself to unauthorised intruders.

Since contributions to an organisation's website can come from diverse sources and departments, it can become difficult to establish accountability and responsibility for the web application. In conventional business software applications it is clearly a particular division or group in the company that is served and that would have an interest in the software working properly. This interest is often diffused and spread out in a web application. It cannot be the technical team or the software wing, as this is often an outsourced vendor. In any case the ownership of the application needs to be assumed by the business and not a technical group. While there can be many claimants for ownership and authority, the same may not hold when it comes to owning responsibility.

To determine who is responsible for the website, some of the relevant questions can be –

- Who pays for the website (this can be a charge to an internal budget)?
- Who is concerned with the accuracy of the content that is published?
- Who gives the requirements to the development or maintenance team?
- Who ensures quality control checks and does the acceptance check?

It is this owner of the website who should be ensuring that proper configuration management practices are followed. Even if this question is resolved, we have to grapple with how exactly to do configuration management. The identification of configuration items can be difficult. This is because there are many components to a web application, and some items such as news content can be very short-lived. Other content can last longer. To what extent does one look at configurations for transient content? Perhaps just recording when it was published and when it was removed is sufficient. For other content that may appear for days or weeks, we might have to deal with normal, conventional changes and updates. What constitutes a configuration unit is also not easy to define, as hyperlinks from within content can complicate the matter.

Conventional configuration management tools are also not really suited to deal with this kind of constant evolution. The usage of these tools is also made difficult because many in the maintenance team, such as content developers, may not be familiar with software, let alone the tools. Also, web application support teams may not be geographically located in the same place and people could be working in different places. So the tools used need to be such that they can be operated over the network.

Project Management

If configuration management has its challenges as described above, they pale into insignificance compared to the difficulties we may have while managing the web

application project itself. When we look at the elements of project management the reasons will become apparent.

1. We have already seen that evolution and hazy, fluid requirement specifications are the norm here. How then does one perform estimation? That presupposes a clear knowledge of the scope and specifications.
2. More than in any other kind of software, schedules are dictated by the users. Quite often, the software is the means to seize upon a business opportunity that may be available for a small window in time. So the developers or maintainers may not have much influence on the time they can get. Whatever the estimate, the software has to be delivered by the date the user needs it.
3. Coordination and human issues can be more daunting than ever. The team may be geographically dispersed, may come from disparate backgrounds and may have conflicting interests or vision.
4. The question of ownership we have already seen in the section on configuration management. Even though stakeholders may not be hard to identify, overall accountability and responsibility could be.
5. The metrics to use to gauge success have to be identified carefully, keeping in mind the unique characteristics of a web application.
6. The actual engineering aspects, particularly analysis, design and testing have to be tuned to the life cycle of such a project. Agile programming methodologies may be of help here.
7. Quality assurance is made more difficult by the frenzied pace of activity that may induce the harried project team to jettison process driven working.

While none of these challenges permit of simple solutions, we can try to keep some things in mind and work towards keeping the project under control by –

- Recognising that the software will follow an iterative life cycle model for development, with each iteration being quite short. We may not be able to work out a grand plan for the whole project as it will be at the end, simply because we cannot predict what shape it will take eventually. So we could concentrate on the current iteration and be clear about what we will deliver in that one. Other items on the user wish list should be taken care of in the next delivery. Save in exceptional circumstances, do not alter the objectives or scope of the current iteration.
- The question of schedule is a difficult one. Managing expectations of the users is imperative here. If what the user's desire is clearly unattainable, we need to negotiate and reduce the scope for the current iteration. Also with more experience in getting a web application developed, the users will hopefully have a better appreciation of what is possible.
- With current day groupware and other tools, we can make it simpler to work with distributed teams. But there should be adequate attention paid to communication among team members so that the objectives of the project are clear. As in a conventional project, we should have team meetings regularly, even if those meetings are done over the telephone or using network meeting tools.
- As always, the processes to be followed in the project should be appropriate to the situation. They should not be cumbersome or difficult to do, so that the project team will be able to follow them. We need to remember that if the processes are complex and are not followed, the impact on the project is much worse, as we potentially could lose control of what is happening.

In subsequent sections we will be reviewing the strategies to follow to get a grip on the other issues such as life cycle activities and measurements to be made. The other, usual elements of project management such as managing risks, analysing defects and

trying to prevent them or monitoring project parameters such as cost, effort, schedule and quality, remain as valid as in any other software project.

Check Your Progress 2

1) The essence of managing the project is proper _____ for the project and then executing the project according to the plan.

2) The _____ process must be up to the demands placed on it of keeping control over the software configuration in spite of the many changes that will happen.

1.4 METRICS

Management of any kind is difficult, if not impractical, without measurements being made. Those values tell us where we stand in our endeavour, whatever it may be. Software engineers have been especially tardy in understanding the need for measurement, preferring to rely on intuition and subjective assessment. However, realisation does seem to have set in and many software projects do measure at least the basic parameters like schedule, effort and defects. When it comes to cost, practices seem to vary across organisations. Some are secretive about the costs of projects and only senior management is privy to such information. Others place the responsibility for the budget for the project on the shoulders of the project manager.

In web applications, these core metrics need to be measured and analysed just as in any other software application. Schedules could be made fine grained so that we are tracking things on small, quick milestones rather than only at the "end" of the project, which is difficult to define here. That could mean that it may not be very meaningful to measure them in terms of slippages (a one-day delay in a task lasting a week is 14%). One possibility could be to look at the percentage of schedules that could not be kept.

Effort metrics can be gathered through a time sheet and we can measure the total effort in different kinds of activities to build up our metrics database for future guidance. However, here we need to be sure to record the actual effort and not try to confine it to a conventional workday of 8 hours. Metrics data on effort is valuable in building up a body of knowledge for future estimation. This data should be recorded under time spent on analysis, design, construction, project management and so forth. Past data is also helpful in negotiating with users for a reasonable amount of time to perform a task

Another core attribute of a software project is its size. This should preferably be measured in an implementation independent way, such as function points. The International Function Point Users Group (IFPUG) has come up with guidelines for measuring the size of a web application from the user point of view. If the organisation has good reason to feel that this measure is not suitable, then the creation of one's own size measure could be considered. If a good size measure can be built, then it can be used to estimate the effort required for future projects, provided the productivity factor that converts the size to effort is available from data on past projects.

The quality of the application can be measured in terms of defects that are found during internal testing or reviews as well as external defects that are reported by visitors who come to the site. But all defects are not equal, and we need to analyse defect data carefully. Some attributes that can be used to classify defects for further analysis are:

- Severity on a scale of 3 levels, such as high, medium and low. More levels up to 5 can also be considered. The severity is in terms of impact on the usage of the software by the user.

- Phase in the life cycle where they were introduced, such as analysis, design or coding.
- Whether they are internal defects, detected by the development or testing team, or whether they were found by the end users.
- Whether they were detected during reviews or during testing.
- Effort required to fix the defect, as this might not have any correlation to the perceived severity of the defect.

Such an analysis would then form the basis for the formulation of a defect prevention plan for the software project. We would strive to prevent the kind of defects that are of high severity or those that take a lot of effort to remedy.

We can also look at other measures like –

- Percentage human resource utilisation in the project. Roles that are idle for significant periods could be allocated to another project as well.
- Percentage of code that is reused from the organisation's library of such components.
- Quantum of changes to the user requirements as the project progresses. An overly high value could indicate that more effort needs to be directed towards scoping out the project and eliciting the requirements.
- Percentage of attrition in the project. An unusually high value for this could point towards problems in managing the human angle.

Besides metrics to give insight into the development aspects of the project and the general state of the team, there can be a large number of statistics and measures that can be gathered about the usage of the website once it has been put into active use. Some of these have been already touched upon in earlier section under the role of Webmaster. Most of them rely on traffic analysis to draw conclusions about the users and their interests. This can give an insight into what changes should be made to the site or how it can be improved.

We can also look at some metrics from the point of view of maintenance. These could be the average time taken to fix bugs of different severity levels, together with the best and worst case times. Other measures can include –

- User satisfaction index
- Mean time between failures (bugs detected or reported)
- Number and percentage of repeat visitors.

Gathering and calculating metrics is only one part of the work involved here. The crux of the matter is how well the knowledge of metrics is used to shape the functionality and features of the project. The metrics should be aligned to the goals and objectives of the project and the lower level measures should be derived from the top level ones, so that we measure whatever is important for us and do not gather extraneous data. From a measurement deficient situation it is quite possible to go overboard and drown oneself in a mass of low level data that is not meaningful or useful.

1.5 ANALYSIS

The usual principles of analysis continue to apply to a web software application as well. So the requirements have to be, as always, elicited, represented and validated. These are then the basis for further work in the project. Let us look at some of the characteristics that we need to bear in mind while analysing a web application project.

It is particularly important to be clear about the process owner and the stakeholders here. We must not end up getting wish lists and requirements from someone who, it later turns out, is not the one authorised to provide them. While always a point of concern, the issues of ownership of the web application accentuate the problem. From the scope that would have been talked about while setting up the project, the detailed system analysis would need to be done to come up with the requirements.

Besides the normal considerations of what the application has to do, we need to look at how the user will interact with the application even more closely than in a conventional application for the reasons we have already looked at in the beginning of this unit. The user interface has to be simple to use, visually appealing and forgiving of a lay visitor who may not be familiar with computers or software. The actual functionality of the application is something that has to be ensured the same way as it would be for any other software, being the very reason for going to all the trouble. Where things are different is in the importance of content to a web application. We have to work out what will be shown to the user, such as a product brochure, an offering for a tour package or anything else. This has to be placed before the user in a manner that will be eye catching but not garish, and the user has to be able to obtain some benefit from the offering without having to do anything elaborate.

More than in conventional applications, it is hard to distinguish entirely between analysis and design in a web application. The kind of actual content that has to be shown needs to be studied to decide on the best way of displaying it on the site. Whether the user will like to show it as text, graphics, audio or video is something that will depend on the resources available for hosting the site as well as on considerations that have to do with the resources available with the group of people that can be expected to visit the site. It is here that the study of the target audience becomes crucial. This should be done on considerations that have to do with –

- Geographical region,
- Language, beliefs and culture,
- Social strata,
- Age group and gender.

and other similar characteristics. All of these demand that we change our design to appeal to our typical visitor.

1.6 DESIGN AND CONSTRUCTION

In designing the functional aspects of a web application, we work much as we would for a conventional application. Because of the compressed time scales and volatility, we should consciously look to reusing as many things as possible. Reuse is always desirable but becomes imperative in the case of a web application. While code and object libraries come first to mind when we think of reuse, there are elements that can be reused for the design as well. We should, for instance, be seeking out any available design patterns that can be reused.

The advent of application servers and other component-based technologies has facilitated the task. The application server gives the framework under which the application can run and takes care of many mundane details that would otherwise be the burden of the programmer. Security and persistence are two such tasks. The functionality of the application is provided by stringing together beans, many of which may have been already written or can be purchased from third parties. There are also other architectural models from other vendors to help write such applications.

Where we have to lay great emphasis is on the user interface. There are several aspects to creating a good one, some of which are:

1. **Visual appeal and aesthetics:** This is the domain of a graphic artist. Things such as cute sound effects, blinking and moving images that can appear interesting or attractive for the first couple of times can irritate after a while. So a pleasant, clean visual design is perhaps a safe bet to hold continuing interest on repeat visits. However, if there is ever a choice between graphical appeal and functionality, we must plumb for functionality every time.
2. **Ease of entry of data:** Given the fact that most users may be unfamiliar with computers or software, data entry should be as simple as can be made. Actual entry using the keyboard should be kept to the minimum. Allow for using the mouse for selection from lists. It should be easy to ask for information of the application.
3. **Ease of obtaining the application's response:** Whenever we query the application, the results should be easy to see and interpret. If the processing is going to take a while, give visual feedback to the user on how long she can expect to wait. At no point should a user feel lost, not knowing where s/he is in the site or what she should do next. Context sensitive help should be available throughout the site.
4. **Intuitive, easy navigation around the site:** As a visitor navigates around the site, there should be some aids to helping her find her bearings. Things like a site map and your location on it are useful. It should be possible to come back to a reference point like the home page in a single click from anywhere deep in the site. It is our application that should provide all navigation facilities and we must not expect the user to use browser buttons to go around.
5. **Robust and forgiving of errors:** If there is any erroneous input, the consequences should not be too inconvenient. We should expect a discreet and graceful error message, not a disconnection from the server. It should not happen that we land up in some error page that does not give us any explanation of what went wrong.

Website users would not like to have to scroll and so we must try to see that this is not required for any screen resolution. A horizontal scroll is much more irritating than a vertical scroll to most users.

1.7 REVIEWS AND TESTING

In this section we will look at some of the considerations for reviewing testing web applications. The content of a website is the item of interest to the visitor and so we must take care to review all content to make sure that there are no errors of fact or language therein. Besides, the design of the site should be reviewed to catch errors like difficulties in navigation or data entry.

It may not be very useful to merely check out the components of a website on a standalone basis as part of unit testing. While in a conventional application a program is the smallest unit that we test, in web applications it is more likely to be a single web page. We should check out the content, the navigation and the links to other pages as part of testing out a page. There should not be any broken links that leave the user bemused. If there is some functionality being provided by that page, that part should work correctly. This part of the software is close to a conventional application and the actual processing that happens can be unit tested as usual. Where the processing is done by using bought out components, it may not be necessary to unit test those components. We would then only have to concentrate on checking out the full functionality.

Once all the pages of the web application are ready we can perform integration testing. We check out how the pages work together. Whenever an error is caught and fixed we should repeat the tests to ensure that solving a problem has not caused

another problem elsewhere. It is possible that individual pages work well but not together.

Unlike a conventional application, here we do not have much control (except our instructions to users) on all parts of the working system. This is because we do not know what browser, what version of it and what operating system the user will be working with. While it may not be practical to test out with all possible combinations, we should ensure that the application works with most sets of browsers and operating systems. We should also see that the software works at different screen resolutions. The network bandwidth expected should also be clearly decided and we should test things out at all network speeds up to the lowest.

The actual implementation of the application has to be done carefully. In the deployment environment, the configuration has to be checked out to be in tune with what the application was designed and constructed for. Thereafter we may release the software to a limited test group of users to gather their feedback before making it public.

☛ Check Your Progress 3

1) The _____ gives the framework under which the application can run and takes care of many mundane details that would otherwise be the burden of the programmer.

2) In the _____ environment, the configuration has to be checked out to be in tune with what the application was designed and constructed for.

1.8 SUMMARY

This unit has been a very brief introduction to different aspects of engineering web software applications. While a lot more could have been written about each topic, and other topics could have been added here, we have been able to concentrate on only a few important points here that are with in the scope.

- Web applications have many points in common with conventional software applications, but have their own unique characteristics.
- There are differences in the ways web applications are developed, deployed, tested and maintained.
    - Web applications often do not have a clear dividing line between development and maintenance and usually continue to evolve.
    - They have to be developed and put into use quickly and so try to make use of reusable components as much as possible.
    - The user interface is very important because they are used by lay visitors.
    - Testing them can be difficult
- Following good management practices like project management and configuration management of web applications is not straightforward because of their characteristics.
- We need to think through the metrics we will use to evaluate project progress and product quality of web applications.

1.9 SOLUTIONS/ANSWERS

Check Your Progress 1

1) Layering approach
2) Concurrent

Check Your Progress 2

1) Planning
2) Change control

Check Your Progress 3

1) Application server
2) Deployment

1.10 FURTHER READING

Software Engineering – A Practitioner's Approach, Roger S. Pressman; McGraw-Hill International Edition.

Reference Websites

- http://sei.cmu.edu/cmmi
- http://standards.ieee.org