International Conference on Systems, Science, Control, Communication, Engineering and Technology 2016 [ICSSCCET 2016]

ISBN	978-81-929866-6-1
Website	icssccet.org
Received	25 – February – 2016
Article ID	ICSSCCET174

VOL	02
eMail	icssccet@asdf.res.in
Accepted	10 - March – 2016
eAID	ICSSCCET.2016.174

Design and Development of Software through Computer Aided Learning for Power Concepts

Geetha K[1], Sudhan J[2], Naveen M[3], Sivajothi B[4], Santhosh Kumar K[5]
[1,2,3,4,5]Department of Electrical and Electronics Engineering, Karpagam Institute of Technology, Coimbatore

Abstract: *Computer aided learning can be done through two ways. 1. Web server installation and 2. Application server installation. Here we are using Application Software installed as an EXE file. In our learning concept we are using MAYA Automation tool for animating characters which are coded with C#. Our gaming idea encloses Sag concept in Power System. The tension of the transmission line can be easily identified by colour variation. The spark gaps and lightning stroke concepts can also be learnt without any statements and pre-defined knowledge.*

INTRODUCTION

A commercial software product is usually derived from market demands. Sales and marketing people have first-hand knowledge of their customers' requirements. Based upon these market requirements, senior software developers create an architecture for the products along with functional and design specifications. Then the development process starts. After the initial development phase, software testing begins, and many times it is done in parallel with the development process. Document actions also part of the development process because a product cannot be brought to market without manuals. Once development and testing are done, the software is released and the support cycle begins. This phase may include bug fixes and new releases. This chapter is not specific to a particular hardware platform or tools.

Concept used for Explaining Software: Voltage sags caused by the short-circuit faults in transmission and distribution lines have become one of the most important power quality problems facing industrial customers and utilities. Voltage sags are normally described by characteristics of both magnitude and duration, but phase angle jump should be taken into account in identifying sag phenomena and finding their solutions. In this paper, voltage sags due to power system faults such as single phase-to-ground, phase-to phase, and two-phase-to-ground faults are characterized by using symmetrical component analysis and their effect on the magnitude variation and phase-angle jumps for each phase are examined.

Life Cycle of a Software Development: Software development is a complicated process comprising many stages. Each stage requires alot of paperwork and documentation in addition to the development and planning process. This is in contrast to the common thinking of newcomers to the software industry who believe that software development is just "writing code." Each software development project has to go through at least the following stages:

- Requirement gathering
- Implementation and coding
- Testing and quality assurance
- Software release
- Documentation

There may be many additional steps and stages depending upon the nature of the software product. You may have to go through multiple cycles during the testing phase as software testers find problems and bugs and developers fix them before a software product is officially released. Let us go into some detail of these stages.

Requirement Gathering

Requirement gathering is usually the first part of any software product. This stage starts when you are thinking about developing software. In this phase, you meet customers or prospective customers, analysing market requirements and features that are in demand. You also find out if there is a real need in the market for the software product you are trying to develop. In this stage, marketing and sales people or people who have direct contact with the customers do most of the work. These people talk to these customers and try to understand what they need. A comprehensive understanding of the customers' needs and writing down features of the proposed software product are the keys to success in this phase. This phase is actually a base for the whole development effort. If the base is not laid correctly, the product will not find a place in the market. If you develop a very good software product which is not required in the market, it does not matter how well you build it. You can find many stories about software products that failed in the market because the customers did not require them.

Implementation and Coding

The software developers take the design documents and development tools (editors, compilers, debuggers etc.) and start writing software. This is usually the longest phase in the product life cycle. Each developer has to write his/her own code and collaborate with other developers to make sure that different components can interoperate with each other. A revision control system such as CVS (Concurrent Versions System) is needed in this phase. There are a few other open source revision control systems as well as commercial options. The version control system provides a central repository to store individual files. A typical software project may contain anywhere from hundreds to thousands of files. In large and complex projects, someone also needs to decide directory hierarchy so that files are stored in appropriate locations. During the development cycle, multiple persons may modify files. If everyone is not following the rules, this may easily break the whole compilation and building process. For example, duplicate definitions of the same variables may cause problems. Similarly, if included files are not written properly, you can easily cause the creation of loops. Other problems pop up when multiple files are included in a single file with conflicting definitions of variables and functions. Coding guidelines should also be defined by architects or senior software developers. For example, if software is intended to be ported to some other platform as well, it should be written on a standard like ANSI. During the implementation process, developers must write enough comments inside the code so that if anybody starts working on the code later on, he/she is able to understand what has already been written. Writing good comments is very important as all other documents, no matter how good they are, will be lost eventually. Ten years after the initial work, you may find only that information which is present inside the code in the form of comments. Development tools also play an important role in this phase of the project. Good development tools save a lot of time for the developers, as well as saving money in terms of improved productivity. The most important development tools for time saving are editors and debuggers. A good editor helps a developer to write code quickly. A good debugger helps make the written code operational in a short period of time. Before starting the coding process, you should spend some time choosing good development tools. Review meetings during the development phase also prove helpful. Potential problems are caught earlier in the development. These meeting are also helpful to keep track of whether the product is on time or if more effort is needed complete it in the required time frame. Sometimes you may also need to make some changes in the design of some components because of new requirements from the marketing and sales people. Review meetings are a great tool to convey these new requirements. Again, architecture documents and MRDs are kept in sync with any changes/problems encountered during development.

Testing

Testing is probably the most important phase for long-term support as well as for the reputation of the company. If you don't control the quality of the software, it will not be able to compete with other products on the market. If software crashes at the customer site, your customer loses productivity as well money and you lose credibility. Sometimes these losses are huge. Unhappy customers will not buy your other products and will not refer other customers to you. You can avoid this situation by doing extensive testing. This testing is referred to as Quality Assurance, or QA, in most of the software world. Usually testing starts as soon as the initial parts of the software are available. There are multiple types of testing and these are explained in this section. Each of these has its own importance.

Unit Testing

Unit testing is testing one part or one component of the product. The developer usually does this when he/she has completed writing code for that part of the product. This makes sure that the component is doing what it is intended to do. This also saves a lot of time for software testers as well as developers by eliminating many cycles of software being passed back and forth between the developer and the tester. When a developer is confident that a particular part of the software is ready, he/she can write test cases to test functionality of this part of the software. The component is then forwarded to the software testing people who run test cases to make sure that the unit is working properly.

Sanity Testing

Sanity testing is a very basic check to see if all software components compile with each other without a problem. This is just to make sure that developers have not defined conflicting or multiple functions or global variable definitions.

Regression or Stress Testing

Regression or stress testing is a process done in some projects to carry out a test for a longer period of time. This type of testing is used to determine behavior of a product when used continuously over a period of time. It can reveal some bugs in software related to memory leakage. In some cases developers allocate memory but forget to release it. This problem is known as memory leakage. When a test is conducted for many days or weeks, this problem results in allocation of all of the available memory until no memory is left. This is the point where your software starts showing abnormal behavior.

Functional Testing

Functional testing is carried out to make sure that the software is doing exactly what it is supposed to do. This type of testing is a must before any software is released to customers. Functional testing is done by people whose primary job is software testing, not the developers themselves. In small software projects where a company can't afford dedicated testers, other developers may do functional testing also. The key point to keep in mind is that the person who wrote a software component should not be the person who tested it. A developer will tend to test the software the way he/she has written it. He/she may easily miss any problems in the software. The software testers need to prepare a testing plan for each component of the software. A testing plan consists of test cases that are run on the software. The software tester can prepare these test cases using functional specifications documents. The tester may also get help from the developer to create test cases. Each test case should include methodology used for testing and expected results. In addition to test cases, the tester may also need to create a certain infrastructure or environment to test the functionality of a piece of code. For example, you may simulate a network to test routing algorithms that may be part of a routing product. The next important job of the tester is to create a service request if an anomaly is found. The tester should include as much information in the service request as possible. Typical information included in reporting bugs includes:

- Test case description
- How the test was carried out
- Expected results
- Results obtained
- If a particular environment was created for testing, a description of that environment.

Branches

In almost all serious software development projects, a revision or version control system is used. This version control system keeps a record of changes in source code files and is usually built in a tree-like structure. When software is released, the state of each file that is part of the release should be recorded. Future developments are made by creating branches to this state. Sometimes special branches may also be created that are solely used for bug fixing.

Release Notes

Every software version contains release notes. These release notes are prepared by people releasing the software version with the help of developers. Release notes show what happened in this software version. Typically the information includes:

- Bug fixes
- New functionality
- Detail of new features added to the software
- Any bugs that are not yet fixed
- Future enhancements
- If a user needs a change in the configuration process, it is also mentioned in the release notes typically a user should be given enough information to understand the new release enhancements and decide whether an upgrade is required or not.

Documentation

1. Technical documentation developed during the development process. This includes architecture, functional and design documents.
2. Technical documentation prepared for technical support staff. This includes technical manuals that support staff use to provide customer support.
3. End-user manuals and guides. This is the documentation for the end user to assist the user getting started with the product and using it.

All three types of documents are necessary for different aspects of product support. Technical documents are necessary for future development, bug fixes, and adding new features. Technical documentation for technical support staff contains information that is too complicated for the end user to understand and use. The support staffs needs this information in addition to user manuals to better support customers. Finally each product must contain user manuals. Technical writers usually develop user manuals which are based on functional specifications.

In the timelines of most software development projects, functional specifications are prepared before code development starts. So the technical writers can start writing user manuals while developers are writing code. By the time the product is ready, most of the work on user manuals has already been completed.

Designing a Game for the Concept SAG Tools Used

Electric Image Animation System

The Electric Image Animation System (EIAS) is a 3D computer graphics package published by EIAS3D. It currently runs on the OS X and Windows platforms.

Components

The Electric Image Animation System is not a single program, but rather a suite of several programs designed to work together. Each of the primary programs handles a particular part of the production workflow:

Animator

Animator is the EIAS animation program. It can directly import 3D models in the Light wave, 3D Studio, AutoCAD, Maya, and Electric Image FACT formats. In addition to animating models, Animator allows you to set up rendering settings. It efficiently supports the animation of very geometrically complex projects.

Camera

Camera is the EIAS rendering program, known for its speed and high image quality. As of version 9.0, it supports ray tracing, Phong shading, scan line rendering, spatial anti-aliasing, motion blur, caustics, radiosity, Photon mapping, Irradiance Cache, Screen Cache and global illumination. Camera outputs to several file formats, like Quick time and EI's own Image format. The latter is directly supported by Adobe after Effects CC and Adobe Photoshop CC.

Animation

The bouncing ball animation (below) consists of these six frames. This animation moves at 10 frames per second. Animation is the process of making the illusion of motion and change by means of the rapid display of a sequence of static images that minimally differ from each other. The illusion—as in motion pictures in general—is thought to rely on the phi phenomenon. Animators are artists who specialize in the creation of animation.

Animation can be recorded with either analogue media, a flip book, motion picture film, video tape, digital media, including formats with animated GIF, Flash animation and digital video. To display animation, a digital camera, computer, or projector are used along with new technologies that are produced.

Animation creation methods include the traditional animation creation method and those involving stop motion animation of two and three-dimensional objects, paper cut outs, puppets and clay figures. Images are displayed in a rapid succession, usually 24, 25, 30, or 60 frames per second.

Tool used for Animating Characters in Transmission Line

Maya is an application used to generate 3D assets for use in film, television, game development and architecture. The software was initially released for the IRIX operating system. However, this support was discontinued in August 2006 after the release of version 6.5. Maya was available in both "Complete" and "Unlimited" editions until August 2008, when it was turned into a single suite.

Users define a virtual workspace (scene) to implement and edit media of a particular project. Scenes can be saved in a variety of formats, the default being .mb (Maya Binary). Maya exposes a node graph architecture. Scene elements are node-based, each node having its own attributes and customization. As a result, the visual representation of a scene is based entirely on a network of interconnecting nodes, depending on each other's information. For the convenience of viewing these networks, there is a dependency and a directed acyclic graph.

Users who are students, teachers (or veterans or unemployed in USA markets) can download a full educational version from the Autodesk Education community. The versions available at the community are only licensed for non commercial use (once activated with the product license) and some products create watermarks on output renders. The software comes with a full 36 month license. Once it expires, users can log in to the community to request a new 36 months license and download the latest Autodesk product.

Maya Embedded Language

Alongside its more recognized visual workflow, Maya is equipped with a cross-platform scripting language, called Maya Embedded Language. MEL is provided for scripting and a means to customize the core functionality of the software, since many of the tools and commands used are written in it. Code can be used to engineer modifications, plug-ins or be injected into runtime. Outside these superficial uses of the language, user interaction is recorded in MEL, allowing even inexperienced users to implement subroutines. Scene information can thus be dumped, extension .ma, editable outside Maya in any text editor.

Supported Operating Systems

Autodesk Maya 2016 is supported on 64-bit Windows (Windows 7 (SP1) or later), Mac (OS X 10.9.5 or later), and Linux (Red Hat Enterprise Linux 6.5 WS or Cent OS 6.5) platforms. Support for Silicon Graphics IRIX was dropped after version 6.5 and opens USE Linux support was dropped in Maya 2009.

Conclusion

Thus Voltage SAG concepts are clearly learnt by using MAYA automation software, Electric image automation system. The transmission lines can be identified by continuous blue, red alternative lines and now the character falls if there present any voltage sag. Finally computer aided learning can made easy and carved for any required concept.