ASSIGNMENT OF CASE TOOL

Name: Muhammad Asim Shahzad

Class: BSSE 6th SELF

Roll No: BSEF17E50

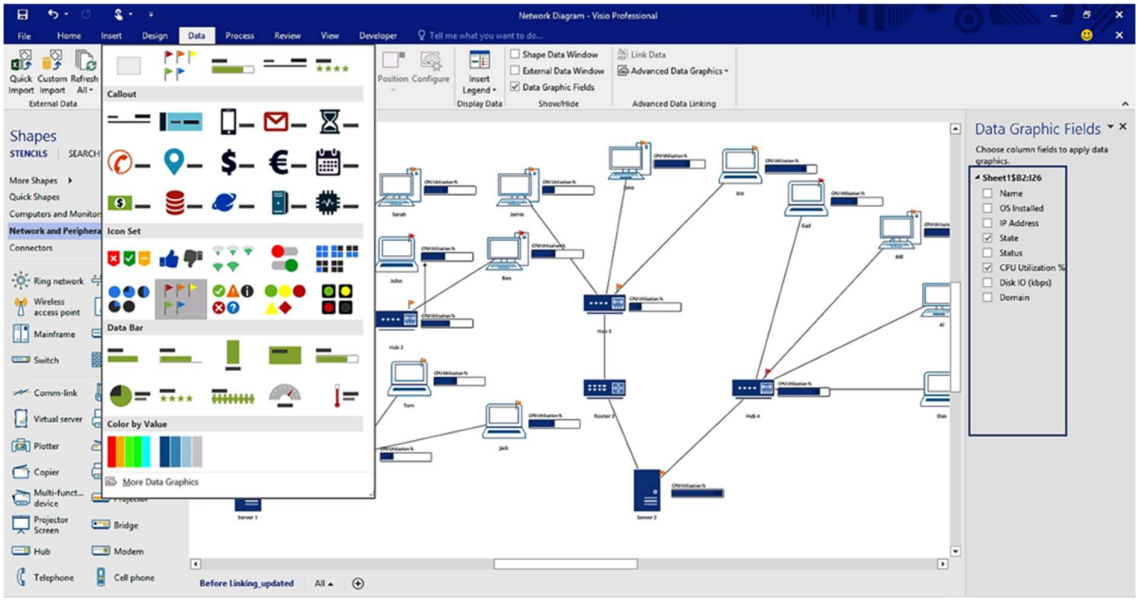
Submitted To: Sir, Ghaffar Khan

Question: Write comparison of Upper, Integrated and Lower-CASE Tools.

Upper Case Tools

i- Microsoft Visio:

Microsoft Visio is an application for diagramming and vector graphics, and is part of the Microsoft Office family. First launched in 1992, the product was made by Shapeware Corporation. Microsoft purchased it in the year 2000.



Pros:

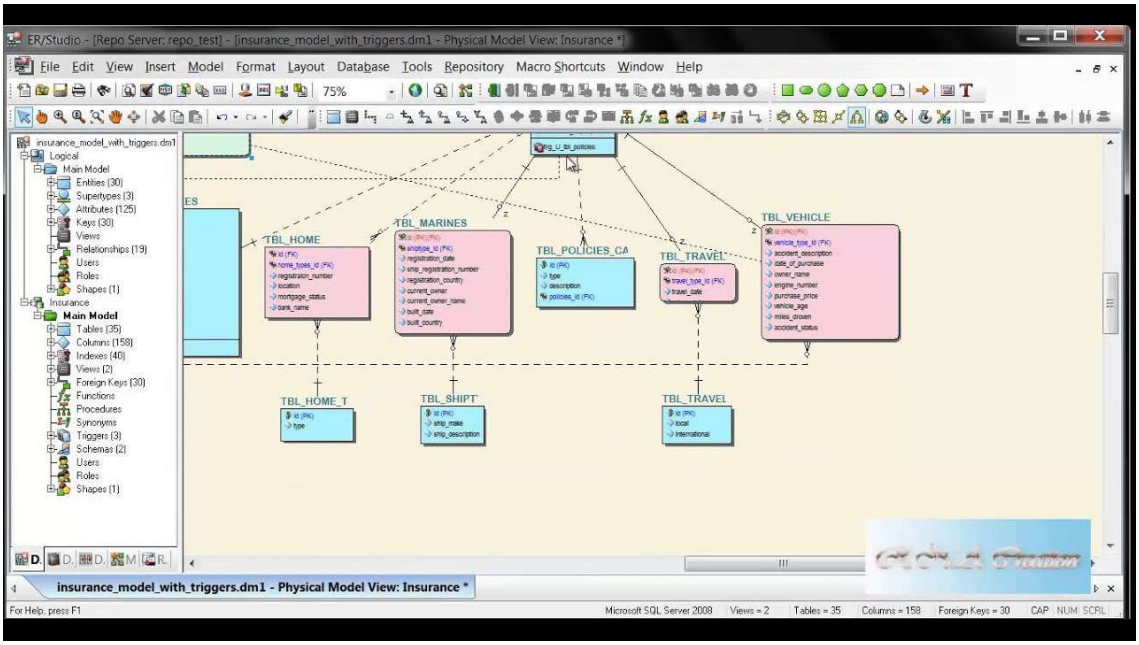
- Easy to find shapes and stencils.
- Built-in templates.
- Powerful and clean visualizations.
- Large library of industry-recognized shapes to leverage in your visualizations.
- Network diagrams.
- Visio is integrated with other Office suite products which is nice.

Cons:

- The software is not intuitive.
- It is quite expensive.
- Sharing your diagram is not easy; printing or publishing is unfortunately not as easy as it could be.
- Keeping shapes aligned could be easier. You have to have a good eye for spatial design to keep complicated diagrams easy-on-the-eyes.

ii- ER/Studio:

ER / Studio is software developed by Embarcadero Technologies for data architecture and database design. ER / Studio is compatible with various database systems and is used to build and maintain database designs, record and reuse data assets by system developers, system modelers, database managers, and business analysts.



Pros:

- ER\Studio supports the concept of a domain. You can create a custom domain (data type) and give it a name and use it as the data type for attributes. I use this to define my surrogate PK and several other "standard" attributes we populate into every relation (table).
- ER\Studio supports the coding of Macros. This is VB code that leverages their API to automate manual process. Very helpful as you could imagine. They supply a large set of pre-canned macros and are very good about helping you write your own if you need help. I have several macros that help me implement naming conventions on attributes and FK names.
- The repository is a handy feature that allows you to save off your models to a database to be safe and also to allow for collaboration between other modelers on you team.
- Several different licensing models allow for flexibility, usage. They support a single user workstation as well as concurrent licensing, for multiple part time users.

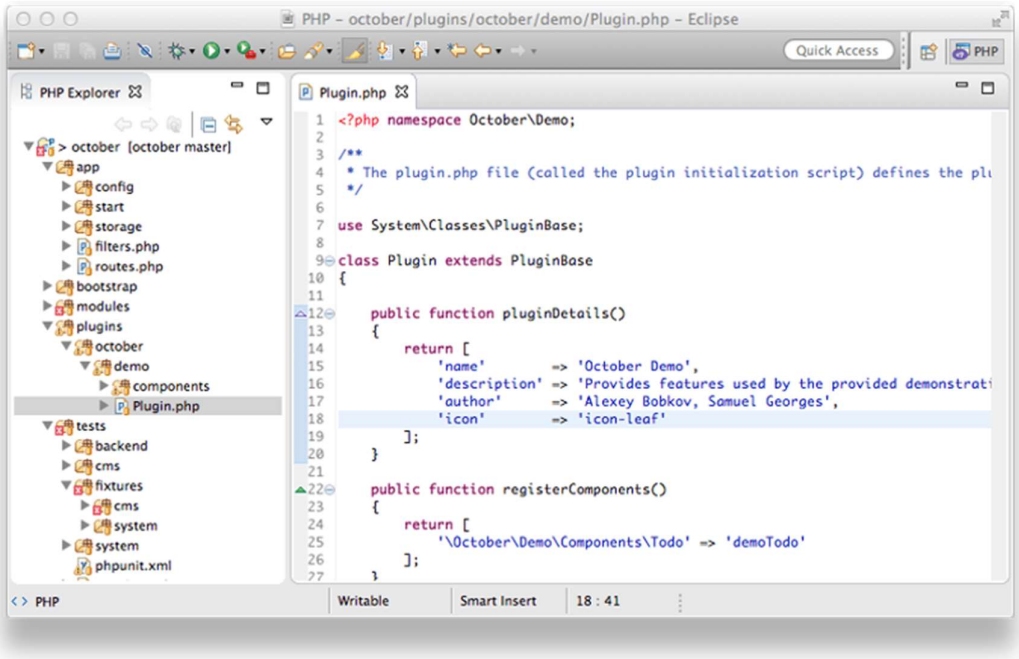
Cons:

- ER\Studio licensing can be cumbersome and upgrading from one version to another usually takes several phone calls and emails to the licensing group to get the update installed and running.
- The repository can be slow when the model count gets larger. By large I mean 20 to 30 models.
- A nice feature that I would like to see is table comments be displayed on the model along with the attributes. Currently you have to choose between the two.

Integrated Case-Tools

i- Eclipse:

Eclipse is an integrated development environment used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment.



Pros:

- It is very good at managing many files under edit. I like the ability to manage multiple projects and multiple files. It supports a wide variety of file formats with type-specific syntax formatting.

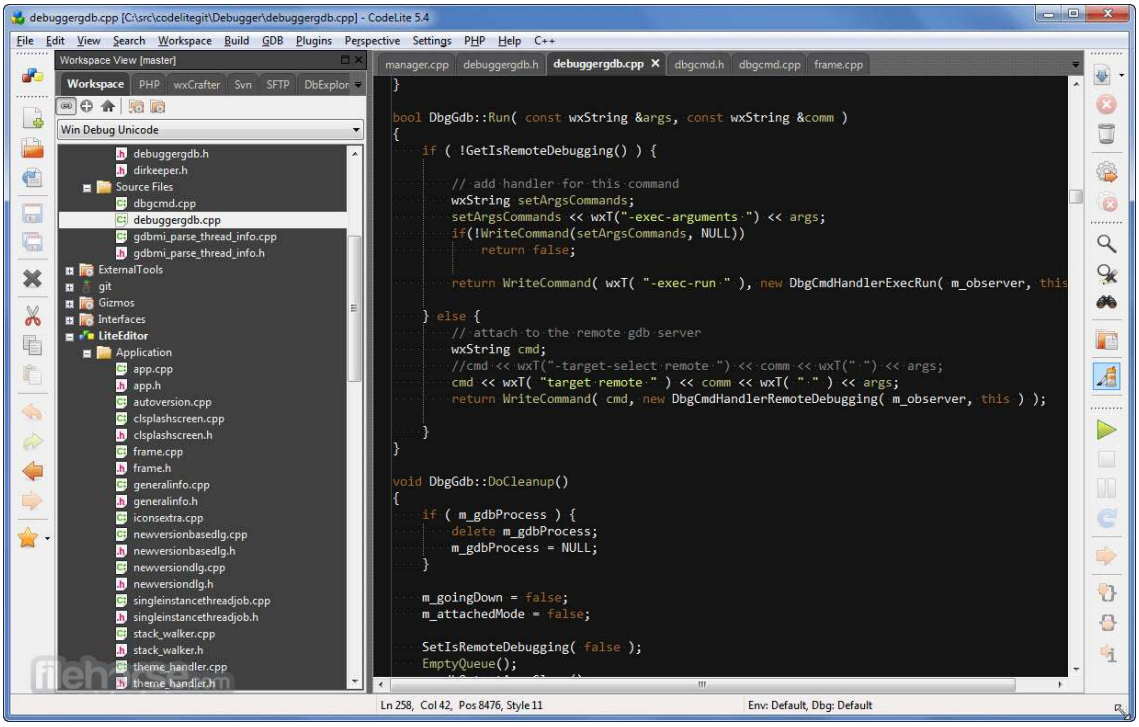
- I like the integrated debugging facility. In particular, we used a remote file system debugger with Python in external VMs to great effect.
- GIT integration is very effective. You can easily manage repositories and connect them to projects, and the project integration into GIT is virtually seamless.

Cons:

- While the DB integration is broad (many connectors) it isn't particularly deep. So if you need to do serious DB work on (for example) SQL Server, it is sometimes necessary to go directly to the SQL Server Studio. But for general access and manipulation, it is ok.
- The syntax formatting is sometimes painful to set up and doesn't always support things well. For example, it doesn't effectively support SCSS.
- The debugging console is not the default, and my choice is never remembered, so every time I restart my program, it's a dialog and several clicks to get it back. The debugging console has the same contextual problems with remote debugging that the editor does.

ii- Codelite

CodeLite is a free, open-source, cross-platform IDE for the C/C++ programming languages using the wxWidgets toolkit. To comply with CodeLite's open-source spirit, the program itself is compiled and debugged using only free tools (MinGW and GDB) for Mac OS X, Windows, Linux and FreeBSD, though CodeLite can execute any third-party compiler or tool that has a command-line interface.



Pros:

- Very fast, easy, with nice design IDE.
- Best cross-platform IDE.
- Great IDE for Linux. Migrating from VC++ I was really surprised to see such a mature product - CodeLite with GCC tool chain. Thanks to intuitive interface it takes minutes to start working. And it has Qt, Git, Subversion, Doxygen, etc. integration options.

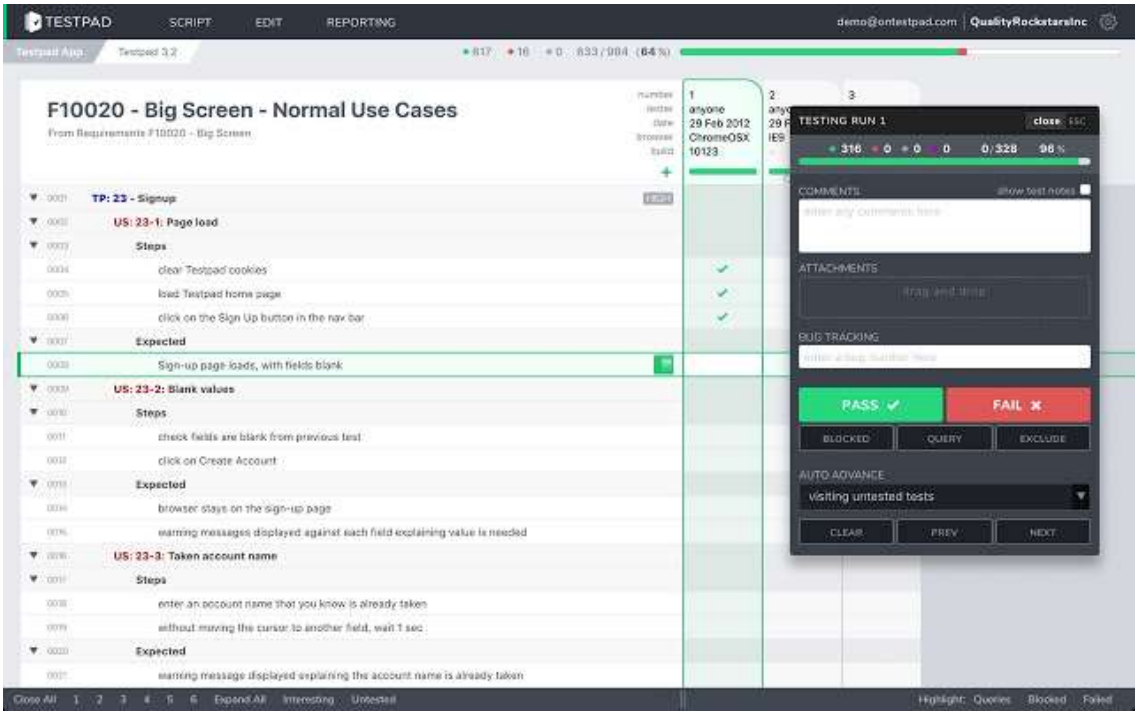
Cons:

- I have had problems of compatibility in the Windows operating system, since sometimes I have unexpected closures and in other computers it can not even be executed, nevertheless in linux I have not had problems.
- Codelite does not let automatically choose the language for you. It's kind of a hassle to have to do that everytime you open up the IDE.
- Since its a free tool can't complain much about it but alsgood the tool is friendly for application developer, the support can be.less expected.

Lower-Case Tools

i- TestPad:

Testpad is a simpler and more accessible manual test tool that prioritises pragmatism over process. Instead of managing cases one at a time, it uses checklist-inspired test plans that can be adapted to a wide range of styles



including Exploratory testing, the manual side of Agile, syntax highlighted BDD, and even traditional test case management.

Pros:

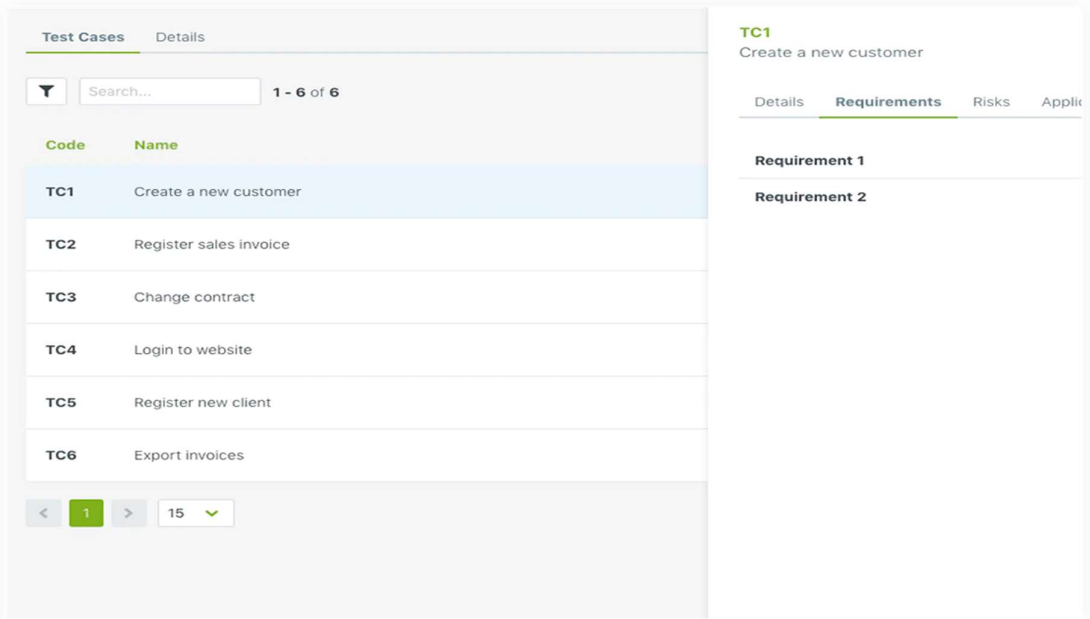
- Guest testers, invited by email, who don’t need accounts
- Simple enough to use by non-testers; get everyone to help at release time
- Keyboard-driven editor with a javascript-powered (i.e. responsive) UI
- Drag'n'drop organisation of test plans.
- Add new tests during testing, as you think of new ideas.

Cons:

- Unfortunately, the interface gets in the way more often than not. The developers insist on overwriting your browser's right-click function, which is something I strongly dislike in Webapps. If a Webapp runs in a browser, it should allow native browser functionality. It feels very strongly as if they wanted to write a Desktop App, but had to settle for a Webapp instead.
- The test needs to be structured to be a either a definite positive or negative, so each 'test' needs to be worded pretty specific. It's good for regression checklist but if a new bug comes up there's not a great way to indicate it.

ii- TestMonitor:

TestMonitor is an end-to-end test management tool for every organization. A simple, intuitive approach to testing. Whether you’re implementing enterprise software, need QA, building a quality app or just need a helping hand in your test project, TestMonitor has you covered.



Pros:

- Requirement and risk-based testing.
- Advanced test case design capable of supporting thousands of cases.
- Robust planning tools with multi-tester runs and milestone cloning.
- Comprehensive result tracking.
- Integrated issue management.
- Smart reporting with many filter and visualization options.

Cons:

- It is manual testing and not automated testing. TestMonitor is integrating with automated testtools, but it is not integrated.
- Java plug-ins are needed for adding printscreens. When Java is not up to date the function for adding printscreens does not work.
- You need to start with base knowledge but that gives you an advance during the project. So actualy isn't that a Con