Project #5 – Virtual Display System (VDS)

Preliminary Architectural Concept

version 1.0

Jim Fawcett
05 August 2012

Purpose of the Project:

Imagine that this is the year 2020. Display technology has advanced to the point that it is feasible to provide an active LED backlit display covering an entire wall, for less than $1000. Using several low-power radio transmitter/receiver pairs, the display can sense the position of any one of a number of pens for drawing and sending commands to a display management system. Assume that this works much like a USB mouse, except that the coordinates transmitted to the controller lie on the board immediately below the position of a sending pen, so the display acts like an electronic whiteboard. Also, by this time text-to-speech conversion can provide high quality output, nearly indistinguishable from human speech, and speech recognition has progressed to the point that, after a brief training period, a user can input text quite reliably via this mechanism.

Virtual Display Functionality

This visual and audio technology should support some interesting collaboration systems, and it is the purpose of this project to explore the possibilities that the display technology will provide. Functions that we expect the display to provide include:

1. Creation of UML style drawings
    a. A user draws a free hand sketch of a small component consisting of perhaps a dozen lines or so, and the display transcribes that into a VISO-like element by matching the sketch with a series of component templates.
    b. Perhaps the user will select the strokes that make up a sketch and tap on the drawing surface near the sketch to activate this conversion.
    c. These components can be dragged around the drawing surface and joined with connectors.
    d. The user can also drag a component from a toolbox to the drawing surface.
    e. A drawing can be saved, retrieved, and edited at any time.
    f. Text notes can be attached in callout style boxes to any element of the drawing, and can be hidden or made visible by a pen-based gesture command.
2. Special purpose charts and visuals, based on user-defined templates can be created, placed anywhere on the drawing surface, edited, saved, and retrieved at any time. These charts show information about a current software development project in ways that are briefly described below, and which you will explore as part of this project.
3. Free-hand sketches of components, diagrams, and charts may be created anywhere on the drawing surface, edited, saved, and retrieved for later use. Free-hand sketches of components may be componentized later by matching to predefined templates, as in #1, above. Evidently this will work well only for relatively simple sketches.

4. HTML documents can be created, saved, and retrieved from storage and placed anywhere on the drawing surface. Text from the document can be selected with a pen and edited with voice commands.
5. Webcam images of collaborators can be placed in resizable windows anywhere on the display surface, and linked through Voice over IP.
6. Text messaging that serves as short term storage of communication between participants, but which can be saved to Collaboration Server storage.

The intent of this configuration is that a user can communicate both verbally and visually with other users and also with the Virtual Display System (VDS) and, through it, with a software collaboration system, in which VDS is embedded. We call this larger system the Virtual Repository Testbed System (VRTS), for reasons that will described later.

Why did we use the term Virtual in Virtual Display System? It is the intent of this (preliminary) concept that the Display System surface can be divided into any number of Virtual Displays, each of which has the entire capability of the VDS, but each virtual display may contain items that are different from the other displays. Each of these Virtual Displays can be placed anywhere on the drawing surface, can be resized, have all of the controls and response mechanisms of the entire VDS, and can be minimized or maximized or share the drawing surface with other Virtual Displays. Thus, each Virtual Display has a control panel, a toolbox, a text messaging area, the ability to host an arbitrary number of webcams, documents, drawings, and charts.

A user might, for example, devote a VDS to each of several projects she is currently working on. Another user might use one VDS to view code dependencies, another to build a package diagram for some subsystem, and a third VDS to view test results from a subsystem he is currently implementing.

Virtual Repository Testbed Server System (VRTS)

The display is augmented with several Virtual Servers that implement a Software Repository, a TestBed, and a Collaboration Server. The term Virtual Server means that the functionality of the server can be moved or replicated with a simple command to any machine connected to the internet, provided that the machine can identify itself as a project machine. The purposes of these servers are:

1. **<u>Primary Repository Server:</u>**
   Holds all of the software, test results, and documents that have been checked in and added to the project's current baseline. We say that these software products are certified, e.g., officially part of the current project. The repository supports check-in, check-out, extraction, and versioning of all products. It also provides an array of analysis tools to measure and report on the quality of individual items.
2. **<u>Primary Testbed Server:</u>**
   Supports testing by running a sequence of tests defined by an XML test configuration file. Each test configuration is run on its own thread, and there may be many test configurations concurrently processed. It is the intent of the Testbed design to support continuous testing of the current baseline as new software components are added.
3. **<u>Primary Collaboration Server:</u>**
   Provides the following services and facilities:
   a. A network of work package descriptions that define the flow and sequencing of all work activities for the project, monitor the status of work in-progress and completed, and measure the resources expended on each task. Each work package is assigned to a single Responsible Individual (RI) has a job description and tangible milestones to measure completion.
   b. Provides a notification mechanism and a status collection system that attempts to enforce the collection of brief status reports associated with each work package, perhaps once a week.
   c. Calendars and schedules. Schedules show, on a time-line, work that has completed with resources expended, and work yet to be completed with milestone dates. When projected on a Virtual Display, any workpackage or milestone can be expanded to show its textual description and status.
   d. Mechanisms for scheduling and controlling meetings coupled through one or more Virtual Displays at each location of the meeting participants.
   e. A Virtual Display to support the activities described in the previous item.
   f. A Wiki that is an integral part of the VDS.

The purpose of the virtuality of these servers is to allow a team (or an individual) to put, on a specified project machine, a replica of a repository, testbed, or collaboration server to support team (or individual) work activities. The contents of these servers are not certified, e.g., not an official part of the project, until such time as a product is extracted from its secondary repository and checked into the primary repository. If several teams each have Virtual Collaboration Servers, their Virtual Displays can be connected into ad-hoc networks within a location and between locations to support the efficient exchange of information between teams.

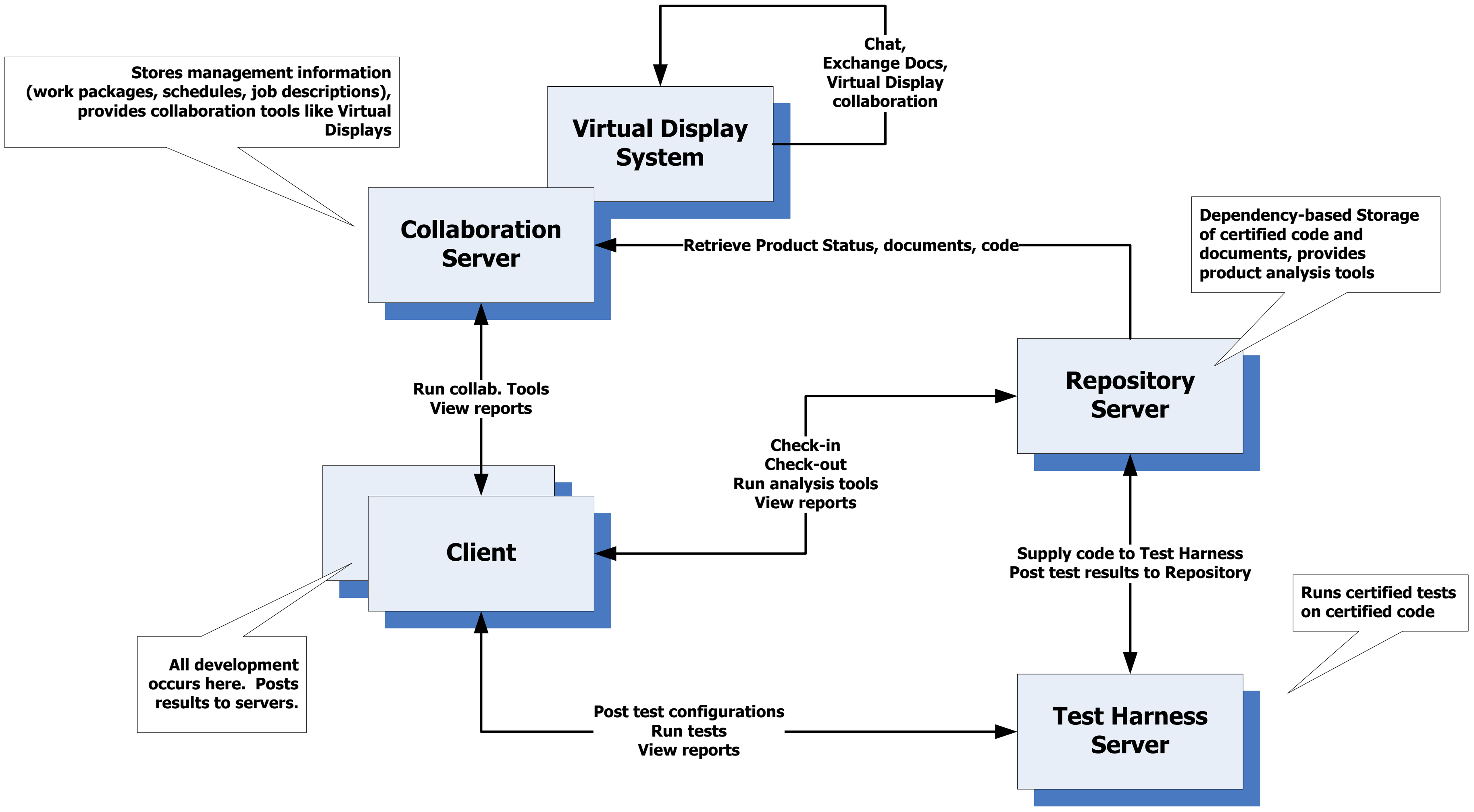
Virtual Display Concept

While all the parts of the VRTS system are interesting, it is the purpose of this project to explore primarily VDS and examine VRTS only for the support it provides VDS. You will do this by developing a concept and architecture for VDS that provides the following features:

1. Virtual Meetings:
   Virtual meetings create, retrieve, discuss, and/or edit any of the items stored on both Repository and Collaboration servers.
   a. These products are to be shared and persistent.
   b. Each item displayed: diagrams, free-hand sketches that have not been componentized, and documents can be either private (visible only on the local Virtual Display) or public (visible on all connected Virtual Displays), and can be saved at any time to a shared store, the Primary Repository, or to unshared store, the local Virtual Repository.
   c. Collaborative drawing and document pasting requires each Virtual Display to send a stream of messages to all other connected displays that represent drawing strokes, documents retrieved from storage, and text messages.
   d. In order to support collaborative interactive drawing, text messaging, and VoIP communication, latency will be an important issue for VDS. Therefore, you should develop a concept for two-tiered storage, e.g., fast short-term storage within VDS, and long-term storage in VRTS Repository servers.
   e. Text messages and VoIP recordings can also be committed to either shared storage (on the Primary Collaboration Server) or local storage (on the local Virtual Collaboration Server).
2. Display Current Project's Software Configuration:
   Show a hierarchy of Systems, Programs, and Packages, connected in dependency order, annotated with names and version numbers, and providing access to detailed information through property sheets, specification and design documents, and source code views.
3. Display all Documents as Web Pages:
   Create, edit, store, and retrieve all documents as HTML pages.
4. Transform all Source Code and Test Results Text into HTML pages:
   Use the services of a code annotator to display all source code as web pages. Use XSLT to transform test data and charts, stored as XML documents, into HTML pages
5. Browsing:
   Provide a simple and intuitive way of navigating through the source code, test results, charts, and diagrams and display them as HTML pages.

6. <u>Work Package Timelines:</u>
   Display a timeline of work packages, colored to show completion status, placed on the time line to show the span of time from start to completion, and clickable to reveal internal details.
   a. The format and structure of this display is determined by an XML-based template.
   b. The time extent of the time line is specified by the user, who may also specify queries to determine what is displayed.
7. <u>Test Results Timelines:</u>
   Display a timeline of test configurations, colored to show pass/fail status, placed on the time line to show the span of time from first test run to last recorded test run, and clickable to reveal internal details.
   a. The format and structure of this display is determined by an XML-based template.
8. <u>Queries:</u>
   Support ad-hoc queries into the Repository or Collaboration Servers databases by selecting tables, and attributes from each table, for display and possibly entering conditions on other attributes that limit the records returned and displayed.
   a. Show tables by tapping on some part of a control panel (what is that?) to get a view of named tables, and tapping on tables to get a view of attributes some of which are selected by tapping.
   b. Get a summary view of what will be returned.
   c. Display the results either in a table or on a time-line.
   d. Results of a typical query might show, in a table, all the tests that failed for a given team between two specified dates, and optionally show them on a time line.
   e. Another typical query might show, in a table, all of the work packages for a given team that completed later than scheduled, and optionally, to display that information on a timeline.
   f. Provide means to optionally save any of these ad-hoc queries as views that could be re-run with a single tap on the control panel at any time.
9. <u>HTML Rendering of Query Views:</u>
   Record an HTML view of any of the queries in #7 and #8 and display on demand.
   a. Provide means to schedule the execution of any of these queries, formatting of the results as a web page, and display in a notices area of the Virtual Display.
10. <u>Common Display of Project Information:</u>
   Provide a Virtual Display, in some common meeting area or areas, which continuously display specific queries in a notices area and supports, in some other part of the display, the execution of new ad-hoc queries. This is intended to communicate to each member of the team, a summary view of the Project's status.

Figure 1. Software Development Collaboration System


Stores management information (work packages, schedules, job descriptions), provides collaboration tools like Virtual Displays
Virtual Display System
Chat, Exchange Docs, Virtual Display collaboration
Collaboration Server
Retrieve Product Status, documents, code
Dependency-based Storage of certified code and documents, provides product analysis tools
Repository Server
Run collab. Tools View reports
Check-in Check-out Run analysis tools View reports
Client
Supply code to Test Harness Post test results to Repository
Runs certified tests on certified code
All development occurs here. Posts results to servers.
Post test configurations Run tests View reports
Test Harness Server

Requirements:

Your **VDS** Architecture document:

1. **shall** be prepared as a Microsoft Office Word file, using embedded Visio diagrams.
2. **shall** partition VDS processing into subsystems. You should partition at a more detailed level for critical subsystems.
3. **shall** explore and describe the user interface(s) you will provide and interfaces between each of the subsystems.
4. **shall** describe the uses/responsibilities, activities, events, and interactions of each of the partitions in your architecture.
5. **shall** use both text and diagrams for the descriptions in 4, above.
6. **Shall** prepare a software prototype of a remote diagramming facility using Windows Communication Foundation (WCF). This prototype is expected to use the diagrammer engine you developed in Project #4 in a remote peer configuration. That is, with two communicating diagrammers, when a diagram is edited in one, the diagram and its changes appears in the display of the other. There are some interesting issues to discuss including preventing the same UI element from being edited simultaneously in multiple remote diagrammers.

You are invited, but not required to prepare additional software prototypes to help you explore the VDS architecture and its issues. Please document prototype code you develop in an Appendix. Note that prototype code should effectively support your conclusions and recommendations concerning the Virtual Display System. Prototypes will be judged by the effectiveness of their design and implementation, the usefulness of its output to convey information about VDS, and the strength of the conclusions you draw, based on the prototype(s).

Additional Prototypes (listed in order of increasing difficulty):

Below find some suggested prototypes. Some of these are fairly simple to implement, some are quite challenging[1].

1. Text to speech conversion
    a. Simple to implement with the FCL Speech Library (look up Speak in the help index).
    b. You will need to do enough to show that Text-to-Speech conversion helps make the user interface more effective.
2. Short-term and long-term storage with streaming communication
    a. Show how to broadcast messages to all connected Virtual Displays and queue the messages in short-term storage at the receiving end for subsequent display. You will need to do a small amount of simple drawing in both sender and multiple receivers to show that your prototype functions correctly.
3. Templates and XML to charts

[1] The more challenging prototypes would make excellent Master's Projects for CE students.

   a. Show how to draw timelines using the vector drawing capabilities of Windows Presentation Foundation (and Silverlight).
4. Dependency graphs
   a. Define a graph class, create an instance that represents some part of the Project code (you can use your Project #1 code to help populate the graph) and plot the resulting graph.
5. Convert visual components to objects and metadata, e.g., how do we convert a drawn component into something we can store in the Repository.
6. Convert free-hand sketches to visual components (as in VISIO).
7. Recognize gesture commands
8. Speech to text conversion

Concluding Comments:

This project statement has provided an initial structuring of the VDS system. You are asked to review this structure and make any revisions that seem appropriate. Then you are to explore the structure and activities of each of the services defined above, and consider any additional services that would improve the usability and effectiveness of the system.

The initial concept is virtually silent about:

1. Suitable user interfaces.
2. The mechanics of file transfer and caching.
3. Details of the communication system.
4. Services provided by Message Handler, Query Handler, and how and when notifications are required.
5. Performance issues associated with collaborative diagramming with the Virtual Display. How many users will VDS support and with what latency?
6. Performance issues associated with database contention.
7. How will users rendezvous to collaborate on documents and diagrams?

There are more questions about these services and about the other services which will be left for you to enumerate and explore.

References:
1. http://www.w3schools.com/xsl/default.asp
2. http://www.learn-xslt-tutorial.com/
3. http://www.zvon.org/xxl/XSLTutorial/Books/Book1/index.html

4. C# High Resolution Timer, based on performance counters, http://www.ecs.syr.edu/faculty/fawcett/handouts/CSE681/code/HiResTimerInterop/