System for designing and creating computer programs using finite state machines

December 29, 2014

Abstract:

The methods commonly used to develop computer software are very difficult, time consuming and error prone. Many software projects are canceled due to cost and time overruns. It is common for projects to fail to deliver all of the anticipated benefits.
What is disclosed is a new paradigm which provides a simple and straightforward method for software development which is much faster and easier to execute. This gain is achieved by using Finite State Machines together with Publish and Subscribe communications utilities. This method can be used in virtually every type of transaction processing environment. It is a good solution for the "Internet of Things".

Keywords:

- Computer Programming
- Software Development
- Object Oriented Programming
- Transaction Processing
- Finite State Machine
- Publish and Subscribe
- Internet of Things
- Database
- Eclipse Rich Client Plugin RCP

Introduction:

The problem being addressed is to devise a method to create robust computer software programs rapidly, reliably, and at low cost. Further, there is a need to maintain the software through its lifetime by having the ability to repair any defects that are reported, and to make any required changes to the system functionality in a rapid and effective manner. Late-arriving specification changes are a reality in every project so they must not be disruptive to the progress already achieved.

The existing solutions involve Relational Database Management Systems as well as Object Oriented Programming methods and techniques. A leading authority has written his opinion that it takes ten years of practice for a developer to become proficient at Object Oriented Programming. This is too long! The current software development process is too demanding, and too stressful. Current trends favor the use of large Enterprise Resource Planning systems that are very difficult to customize to meet individual needs. The industry often speaks of "managing expectations", and seldom of "exceeding expectations". This ensures that every client suffers disappointment.

What is disclosed is a design method in which the steps required to create reliable and flexible software are easy to learn and easy to execute. The new system is "transparent" in the sense

that stakeholders who are not trained in the computer arts can participate in a very fundamental way to specify, define and create their desired software solution. We anticipate that our invention will have a major positive impact on the software industry.

Description:

What is disclosed is a software design strategy that employs methods and concepts that are very old and well established, including:

- Publish and Subscribe of XML Records having prescribed data fields
- TCP/IP networking using socket pairs
- Finite State Machines that are defined by State Tables {State – Events – Transition – Next State}
- Rich Client Plugins that contain a limited amount of functionality and can communicate only by Publish and Subscribe of DataStream records. Rich Client Plugins can employ any operating system and any programming language. Rich Client Plugins may have a Graphical User Interface (GUI), or not. Rich Client Plugins can run in any type of computer hardware such as a desktop PC, server, laptop, tablet, cell phone, embedded microprocessor, etc.
- Local databases are contained within the Rich Client Plugins and are never shared with other Rich Client Plugins.

Hollerith Punch cards were used for the 1890 US Census. In the 1950's, data was stored on 80-column punch cards. Our solution returns to that data encapsulation concept, by using XML records with prescribed fields. These are called DataStream records. They are communicated using Publish and Subscribe methods. Every DataStream record that is published is recorded in a Write Once Read Many store, and there will normally be more than one Write Once Read Many store located far apart for safety and security. Published DataStream records are indelible and cannot be corrected or altered – similar to punch cards.

Networking of the DataStream records is provided by a Broker and an Archivist.
A Librarian is made available to support queries of old DataStream records in the Archive.

This invention is superior to current methods because the required computing functionality can be created and deployed much faster. Late-arriving change orders are not disruptive because they are accommodated by altering one or more Rich Client Plugin and adding new fields to the DataStream records. The operation of every Rich Client Plugin is defined and specified by a Finite State Machine state table that uses this format:

{State – Events – Transition – Next State}

This type of specification document is capable of capturing the requirements in every detail. Because it does not require training in any computer language, all stakeholders can participate in the design process. Further, it is now possible to plan at a high level and to debate the best solution strategy. Design competitions become practical because competing Rich Client Plugins can be created inexpensively.

Steps to Create the Invention:

Using our method, the first step in approaching any new computing project is to design the XML DataStream records and their data fields. These fields must record all that is “of interest”

in the application.

Then the required Rich Client Plugins should be listed, and defined using a separate Finite State Machine table to govern each Rich Client Plugin. The test conditions are specified, and test DataStream records are always provided. Now the Rich Client Plugin can be assigned to a developer. Each Rich Client Plugin should contain a small amount of computer code. This small size promotes good developer comprehension and makes errors less likely. When completed, each Rich Client Plugin can be tested and accepted as complete.

Several developers can work on multiple Rich Client Plugins in parallel if required, in order to speed up the work. These developers can be widely separated geographically and need never meet or converse with each other. The developers are engaged in tactics, not strategy. The strategy was debated and specified by the authors of the State Tables. This is new and refreshing.

The required communications infrastructure components include the Broker, Archivist, and Librarian.

The **Broker** is a simple message Router, which transmits DataStream records to each Subscriber that has registered an interest in a type of DataStream record, and it does this in near-real-time.

The **Archivist** is a Write Once Read Many storage service which saves every DataStream record indelibly, forever. This can be of evidence quality, and serve as highly reliable testimony for forensic analysis.

The **Librarian** is a Query service which returns DataStream records that qualify for each particular query request.

The Rich Client Plugins are examples of generic Logic Boxes (LB) or “Objects” (per Object Oriented Programming) and they are permitted to communicate by a single method alone: DataStream records via Publish and Subscribe to the Broker. A Rich Client Plugin can employ any operating system and any language and any database – these are freely chosen by the developer.

Business Intelligence functions can be performed by the use of commercial Business Intelligence Dashboard products, and these are fed data via a Rich Client Plugin that performs the extract, translate and load services required by the dashboard.

Reports and analysis can be done anytime, even using methods that were never contemplated at the time of the original software design.

Example:

Secure door entry system. The DataStream records and their fields are shown in Figure 2. Rich Client Plugins would be defined to interface with the door locks and the payroll department.
In operation, door Rich Client Plugin would publish the arrival of an employee RFID card Payroll Rich Client Plugin would subscribe to all RFID card messages

Payroll would check its internal database to see if entry is allowed and publish the result
Door Rich Client Plugin would subscribe to the result DataStream record. If allowed, it would unlock the door.

References:

US 395781 - Punch card counting machine, Herman Hollerith, 1887.
Modeling Software with Finite State Machines: A Practical Approach ISBN-10: 0849380863

Modeling and Evaluation of High-performance Publish-Subscribe System - Computational Intelligence and Design, 2008. ISCID '08. Pages 457 - 460

Diagrams:

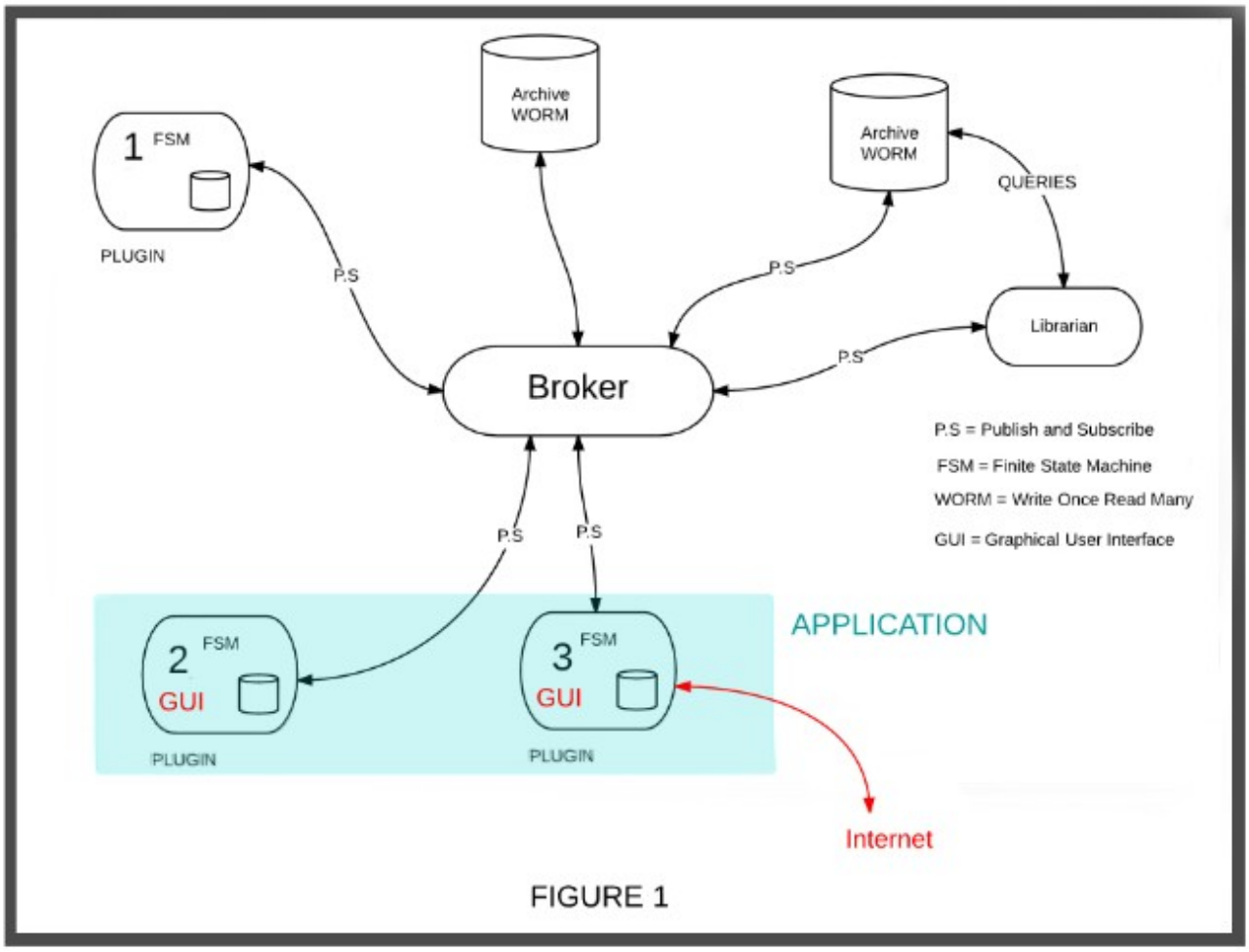

FIGURE 1

RECORD #	FIELD 1	FIELD 2	FIELD 3	PUBLISHER	SUBSCRIBERS	NOTES
DS1	DOOR ID	RFID #		DOOR	PAYROLL, ARCHIVE	Door announces that an RFID has arrived
DS2	DOOR ID		LOCK COMMAND: Unlock / Lock	PAYROLL	DOOR, ARCHIVE	Payroll responds: Unlock or Remain Locked
DS3	DOOR ID	DOOR SWITCH STATE: Open/Closed		DOOR	PAYROLL, ARCHIVE	Door reports that door switch has changed state to Open or Closed
DS4	DOOR ID			PAYROLL	DOOR, ARCHIVE	Payroll interogates Door as to state of Door Switch

FIGURE 2

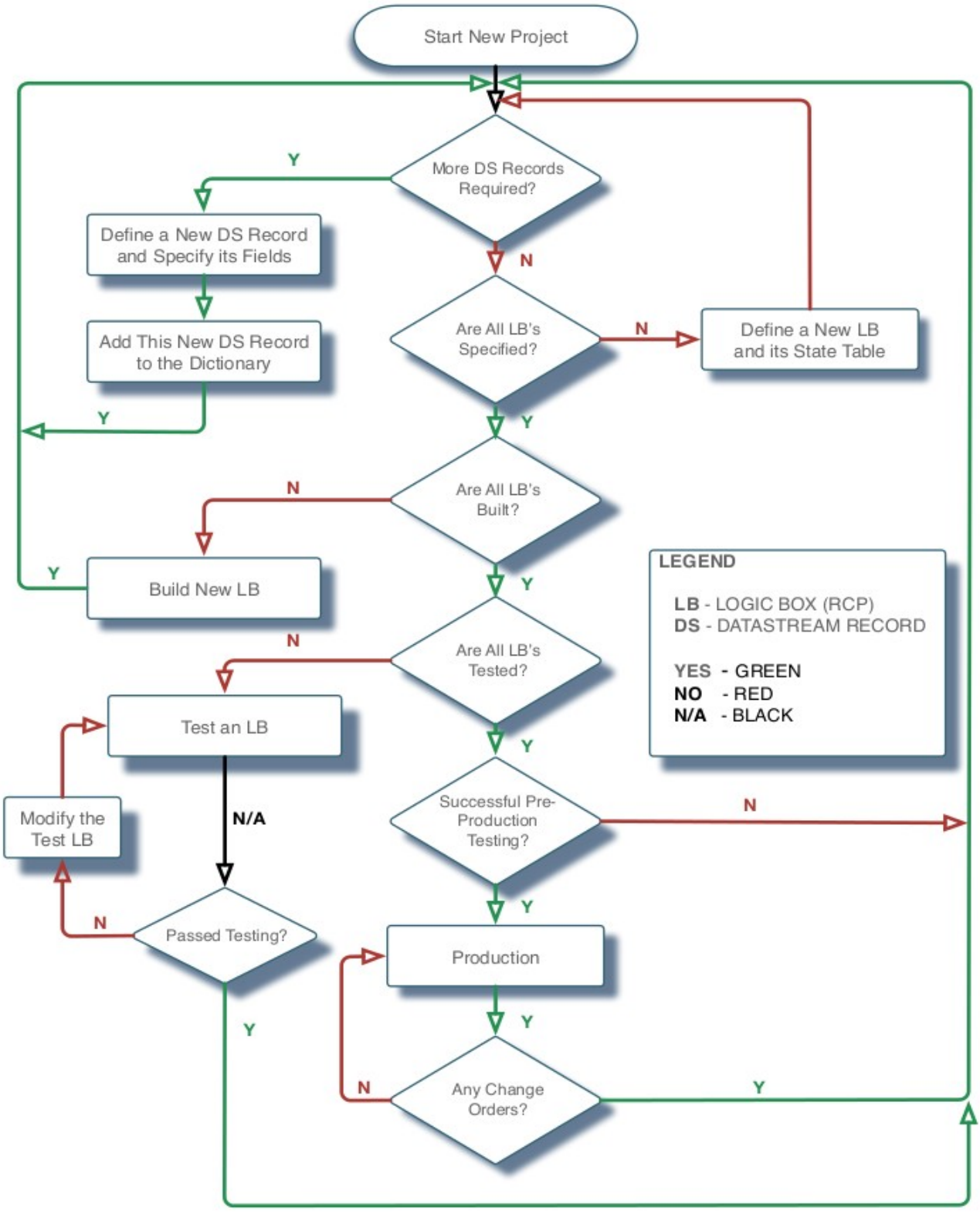

FIGURE 3

FLOWCHART OF PROJECT MANAGEMENT