SSQA Seminar Series

Server Side Testing Frameworks

Sachin Bansal

Sr. Quality Engineering Manager

Adobe Systems Inc.

February 13th , 2007

Agenda

- Introduction
- Drivers for Server Side Testing
  - Challenges in Server Testing
  - Identifying and designing modular Server testing systems
- System testing
- 3 Case Studies/Demo of Automation Systems
- Central Quality center system with Flash and Flex
- Interactive Session (ask questions)

User Interface Testing Vs. Server Testing

- UI Automation
  - Upper layer, indirect way of testing server logic
  - Scripts are Sensitive to cosmetic UI changes
  - Very Slow
  - May not be usable until the UI is complete
- Server Automation
  - Emulate Client requests, repeat UI validations
  - Access by APIs, deal with ‘crazy’ client requests
  - Can be run as unit tests or system tests
  - Very fast

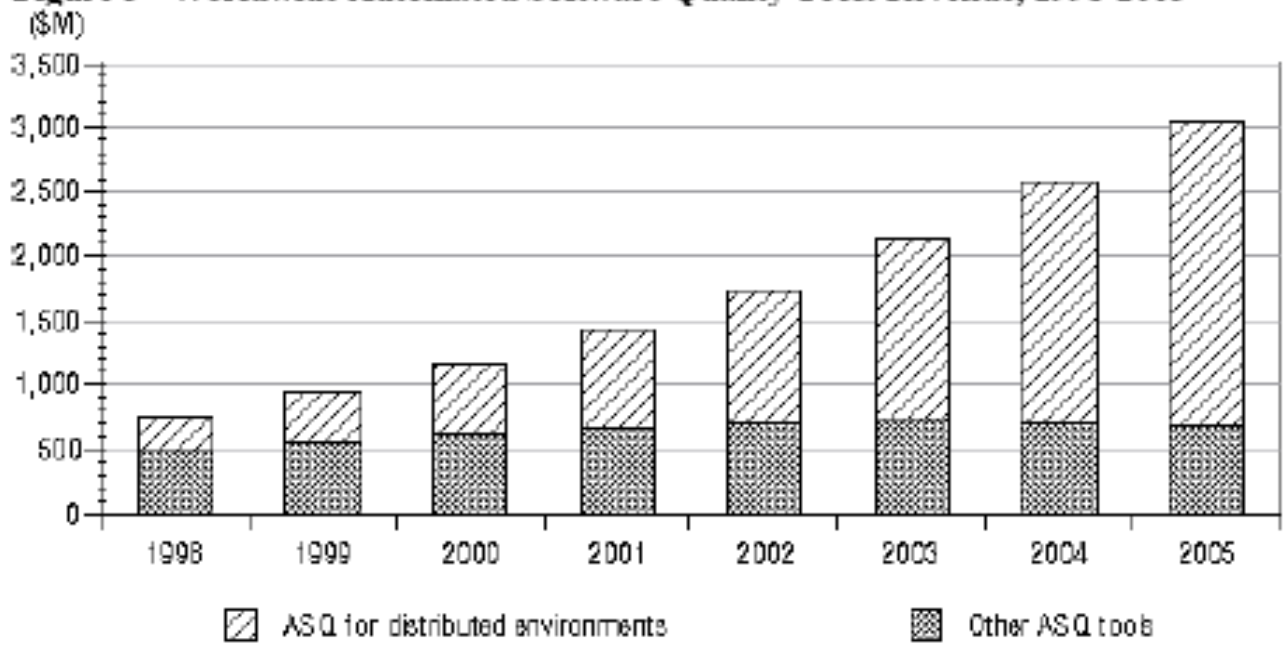
World Wide QE Tools Spending

Table 1 - Worldwide Automated Software Quality Tools for Distributed Environments Revenue by Vendor, 1998-2000 ($M)

	1998	1999	2000	2000 Share (%)	1999-2000 Growth (%)
Mercury Interactive Corp.	112	173	271	50.8	57.0
Rational	61	74	106	19.9	42.3
Segue Software	34	37	46	8.6	23.0
Compuware Corp.	21	29	40	7.4	38.8
Empirix	1	10	22	4.2	124.2
RadView Software	0	5	10	2.0	116.7
Cyrano Corp.	11	9	7	1.2	-29.1
McCabe & Associates	2	3	3	0.6	-7.3
Telcordia Technologies	3	3	3	0.5	-1.4
Computer Associates Intl. Inc	2	4	2	0.5	-32.3
IBM	3	3	2	0.4	-14.5
Hewlett-Packard	2	2	2	0.3	-1.0
Telelogic AB	0	0	1	0.1	76.8
Subtotal	**250**	**351**	**514**	**96.5**	**46.4**
Other	23	22	19	3.5	-16.1
Total	**274**	**374**	**533**	**100.0**	**42.7**

QE Tools Revenue

Figure 5 - Worldwide Automated Software Quality Tools Revenue, 1998-2005



Top Reasons Test Automation Fails

- Not viewing test automation as a resource-consuming project
- Buying the wrong test automation tool
- Using capture/playback as the primary means of creating test cases
- Writing isolated scripts
- Using poorly designed frameworks
- Inadequate test tool training
- Viewing automation as a full replacement for manual testing
- Trying to automate everything - not showing value
- Lack of management support

Challenges in Server Test Automation

- No ideal solution (no – one size fits all).
- Analyze responses, logs (No GUI/visible errors).
- Complex interactions with 3rd party systems
- Complex Workflows
- Lot of Scripts (debugging may not be easy)
- Complex interplay of parameters
  - Network, cache, object pooling, app server configuration, server performance, databases
- When stable, should be able to run as “black box”
- Maintenance

Requirements - Server Test Automation Frameworks

- Act as a generic test platform
- By-pass GUI
- Customizable (configure) and modular (plug and play)
- User friendly, execute in black box manner by anyone
- Reliable (no false positive/negative)
- Should be able to handle all configurations, regressions
- Easy to maintain, easy to debug
- Easy to Install, uninstall, execute, interpret results
- Robust error handling

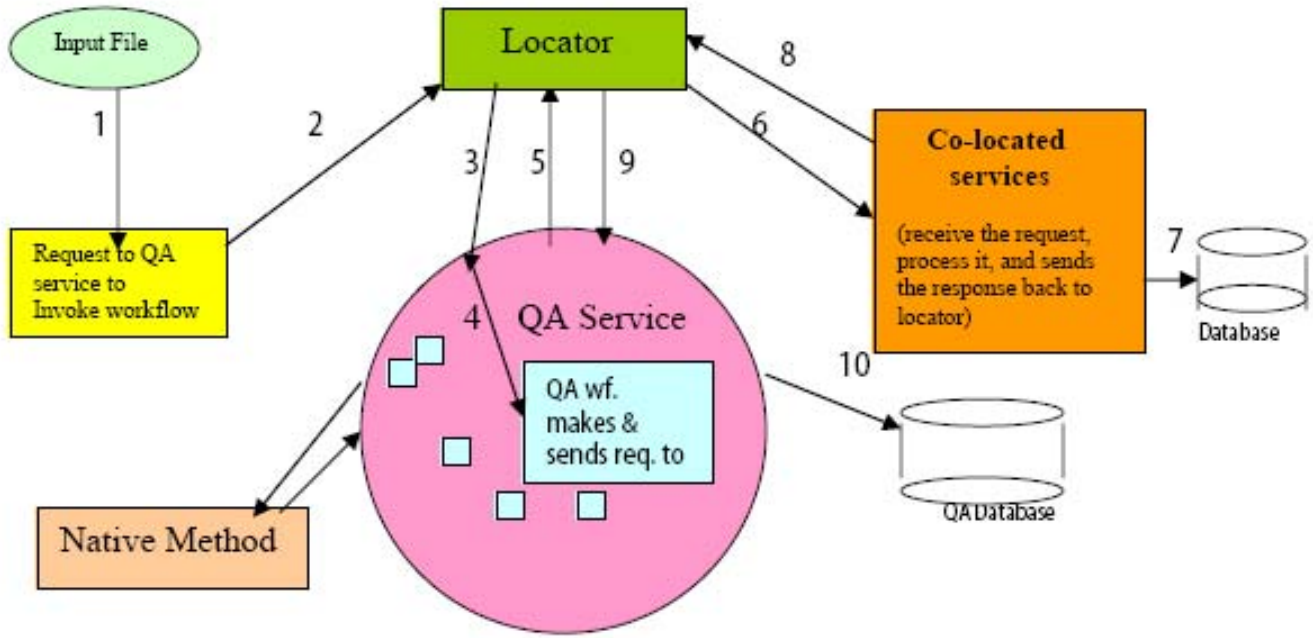
Building block by block – Product Design

- Identify requirements (pain points in manual workflow)
- Design (talk with domain experts)
- Non-intrusive (like monitoring service)
- Implementation (use convenient technologies)
  - Multiple working milestones
- Execution (work with manual QE as it's users)
- Real time Reporting (high level)

Case Studies

- Distributed Quality Server
- Barcode Fill-in Test Automation
- Barcode Form Creation Automation
- LiveCycle QPAC automation

Distributed Quality Server Architecture



Reasons for QA Server

- The QA service set up can be on any local machine and can be connected to any “stack” running remotely.
- QA service does not have to be a part of the build. Problems in the QA service does not effect the rest of the development.
- QA database and tables are isolated from the production database
- QA service has XML based APIs same as other services. It posts a request to the locator with the name of the service which needs to execute the request.

Barcode Form Instance

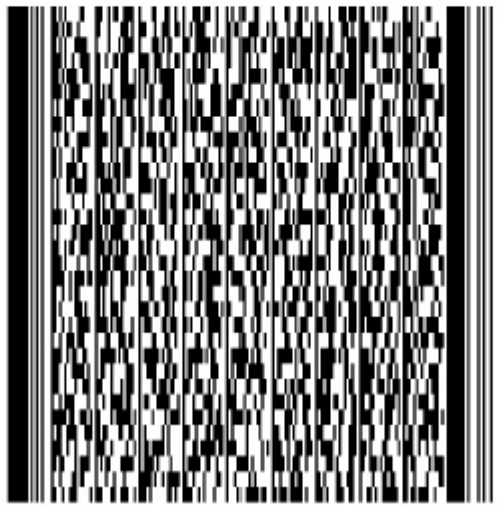
Text Field

Text Field

Text Field

Text Field

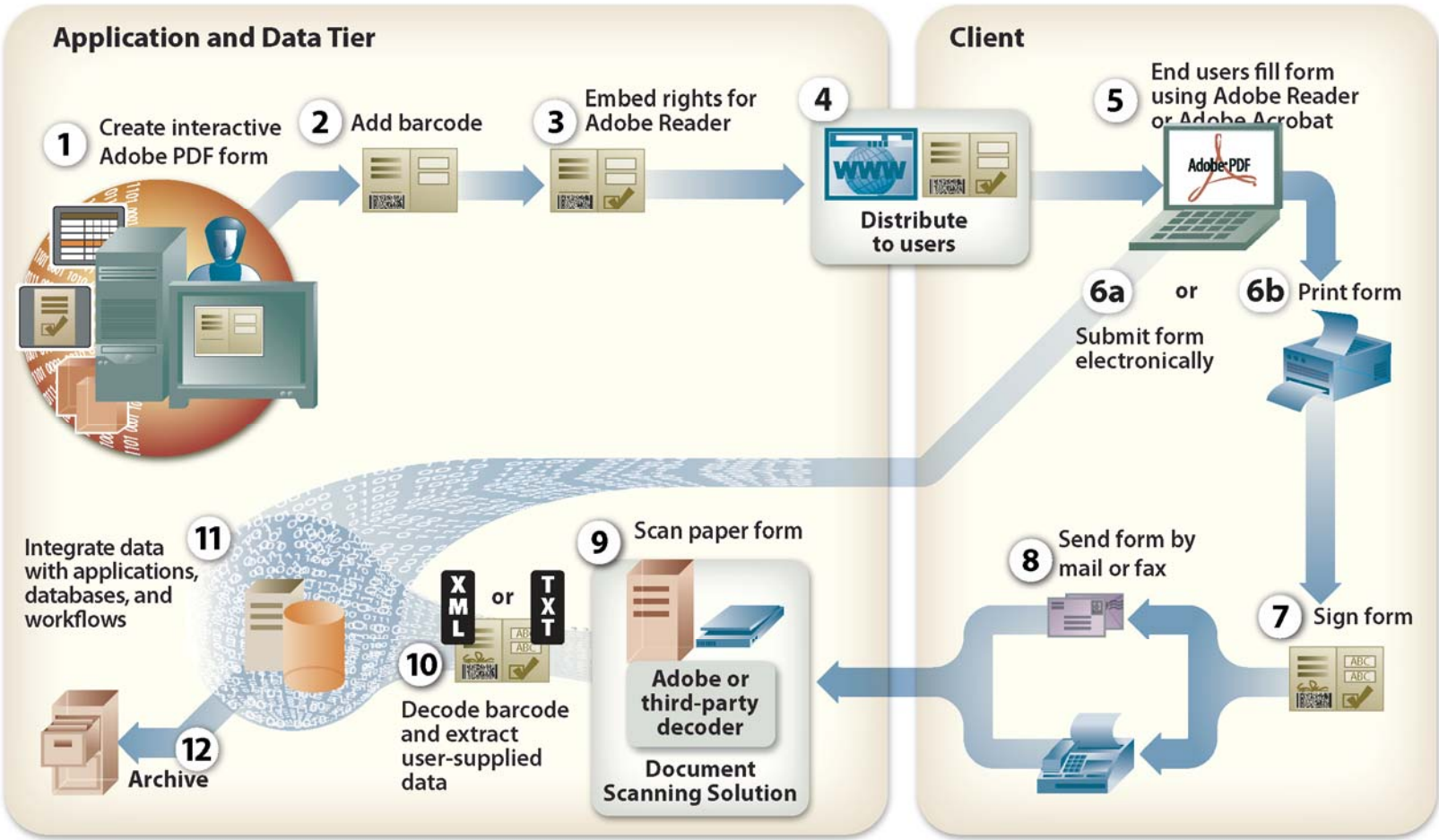
Data Matrix Barcode



PDF417 Barcode

QR Code Barcode

Workflow: Typical Barcode Workflow



Challenge: how to imitate this whole workflow (mostly manual)

- **Encoder Test (client testing):** Byte-by-Byte comparison of what your intent to put in barcode and what got encoded in barcode
- **Decoder Test (server testing):** Byte-by-Byte comparison of what got encoded and what you get after 'decoding' barcode.

Parameters: challenging and complex !!

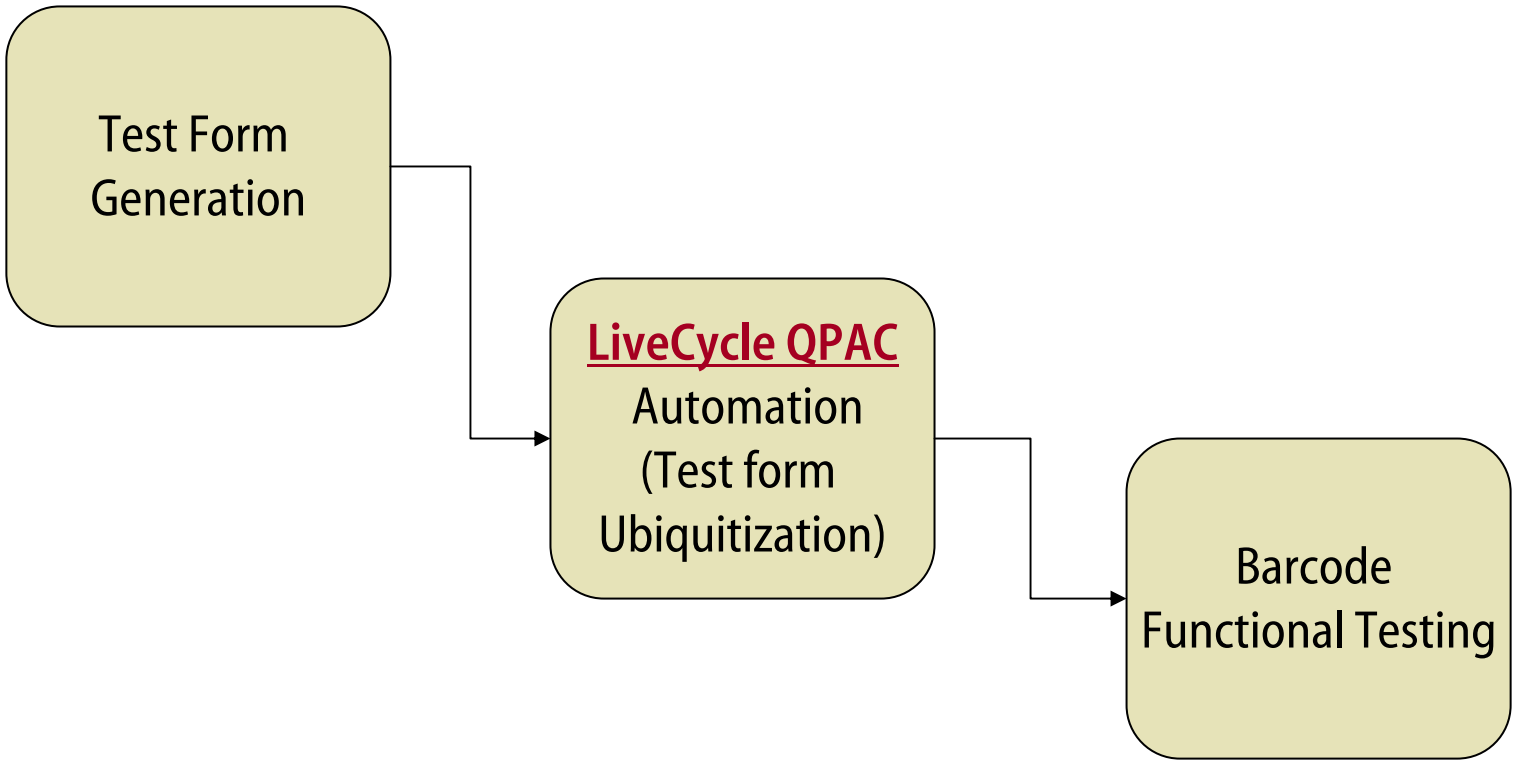
- Interactive Forms and XFA forms with different Ubiquity rights (Barcode, Save).
- Fill the form using either Acrobat or Reader.
- Tests the international/UTF-8 characters.
- Encrypted, compressed barcodes
- Different types of barcodes (called Symbologies ex. PDF417, QRCode, Datamatrix).
- Different versions of Acrobat/Reader and different OS versions, different OS languages.
- Decoder Server, workflow server.
- Decode both PDF/TIF files.
- Semi-Hardware solution (working with a USB key and vendor)

Each build has to be tested for…..

- Every version of Adobe Reader (6.x, 7.x, 8.x)
- Every version of Adobe Acrobat (6.x, 7.x, 8.x)
- Different created and filled version of Acrobat
- All Windows OS (Win 2k, Win Server, XP, Vista)
- All Acrobat languages (German, Japanese…..)
- All Windows languages…..
- All Mac (OSX, MacTel, PowerPC…)
- Cross Language (Acrobat French on German OS)
- Create with Acrobat Filled with Reader/Acrobat

Workflow: Putting it together

- Test Workflow, using various modules

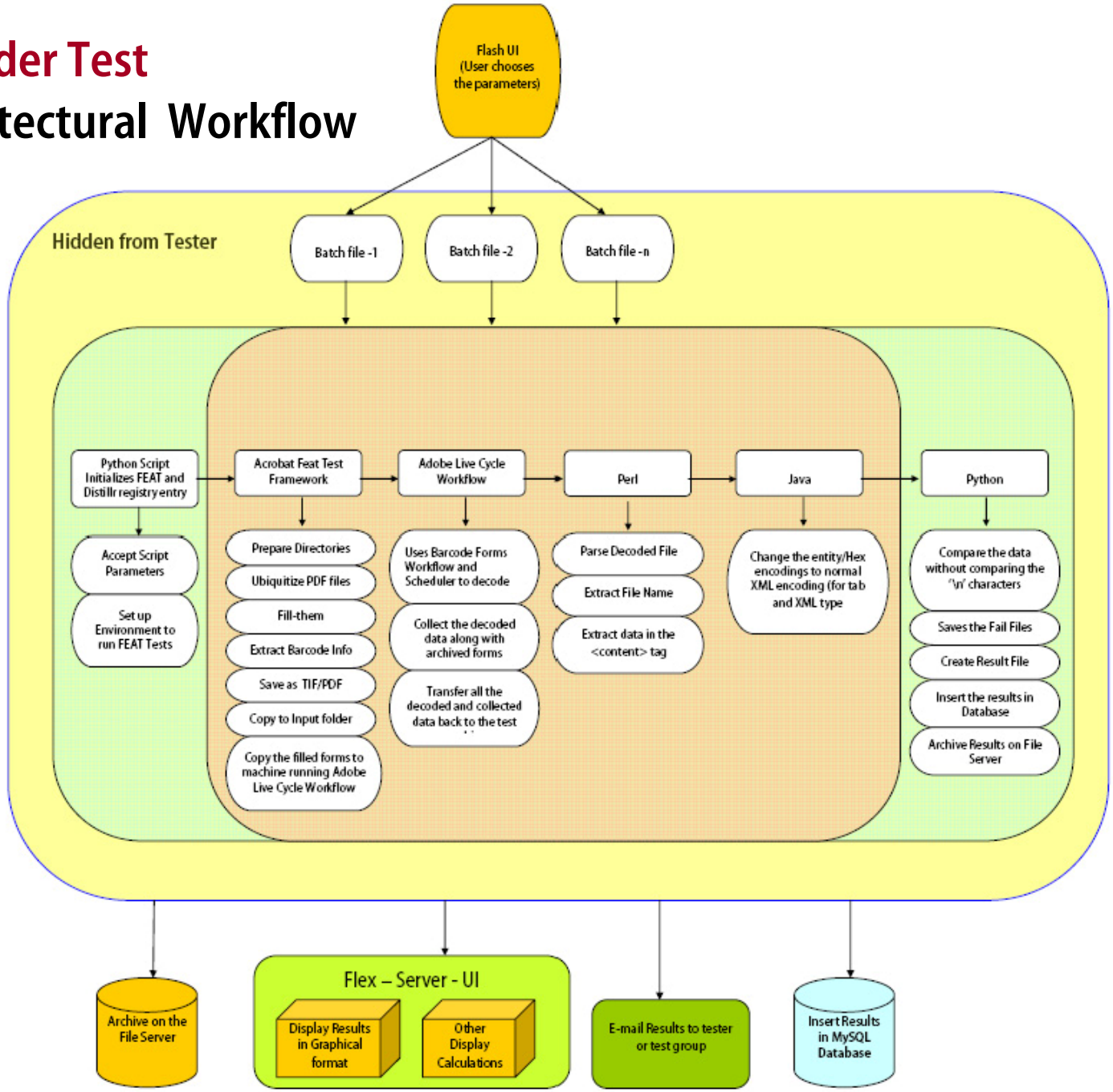


“Deliverables”

- Easy to install and uninstall (has an installer and uninstaller)
- Easy to Use and Run (UI driven)
- Easy to Configure (easy scripting interface if needed)
- Easy to Interpret Results (UI to display results, database to capture result and parameters).
- Easy Archival of the test data (test results and data archived on a file server of your choice)
- Robust Error handling in case of invalid and improper files.
- No ‘false-positives’ and ‘false-negatives’ results.
- Once the tests are run, sends the e-mail with URL of results and details.

Decoder Test

Architectural Workflow



Barcode Form Testing: Current Workflow

- Start the test run using Flash UI.
- Automated filling of Acrobat (acroForms)/Designer forms.
- Save the filled Data in UTF-8 form in text file.
- Save the form as Tiff file (Acrobat) or .ps (Reader, use distiller to convert to pdf).
- Sent PDF/TIFF file to the “Live Cycle Workflow Decoder Server”.
- Decode the form, get the results back to the test machine.
- Parse the result and compare the data with the “filled Data”.
- If the comparison passes, move on to the next file, if comparison fails, save the failed files and move to next test.
- At the end of all the test files, compile the results.
- Save the data in the database.
- Archive the results on a file server.
- Display the results using Flex UI.
- Send the e-mail notification.

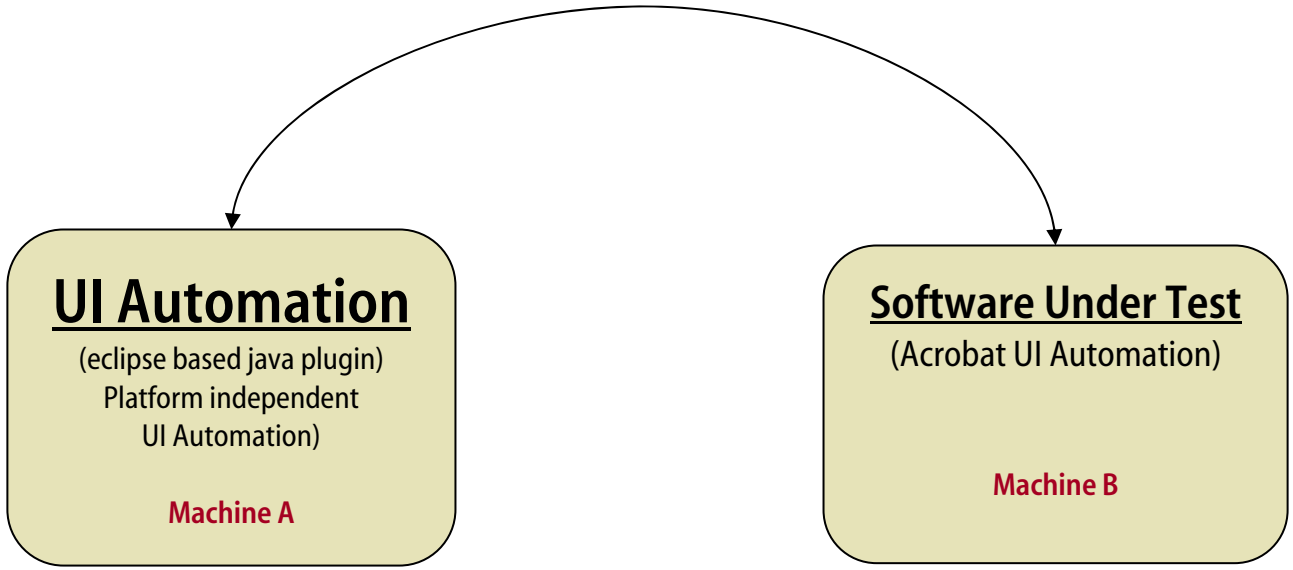
Overview of Technologies Used

Installer is used to install the required languages and tools.

- Flash UI based user interface
- Java JRE 1.4.2_08
- Python 2.4 (with various post install python modules)
- Perl (ActiveState Perl 5.8)
- Acrobat TEST plugin
- Flex: display results of test runs
- MySQL database to store test results

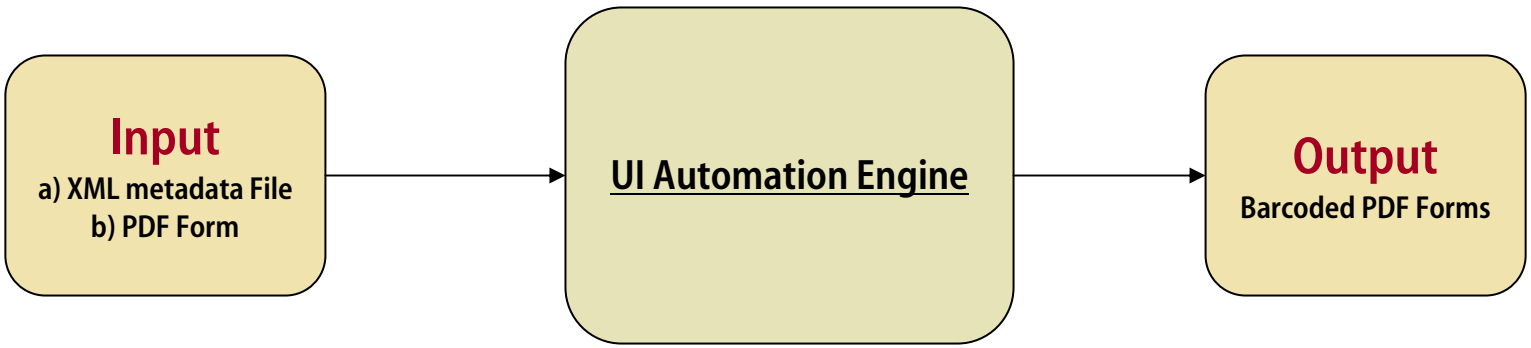
Test Form Generation (test files)

- Architecture



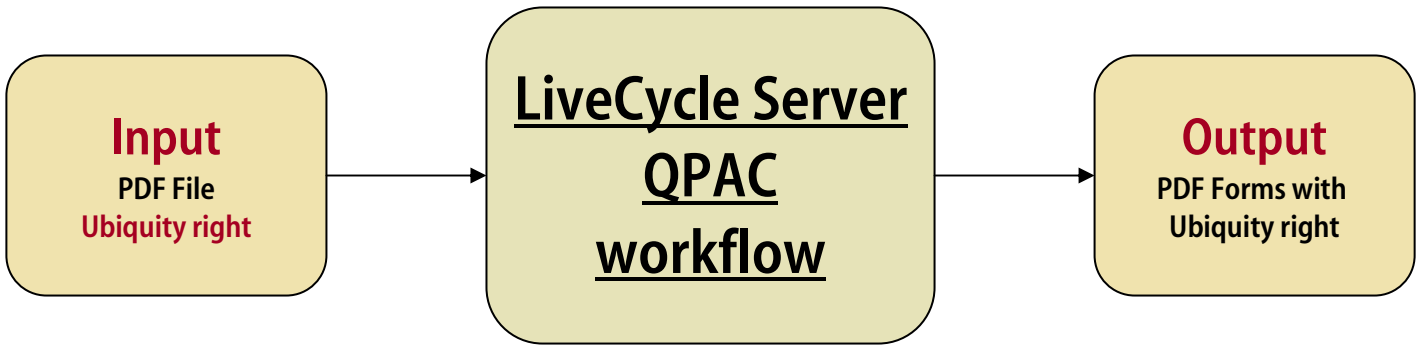
Form Creation Workflow

- Test form Generation Workflow



LiveCycle Server QPAC Workflow

- Test form Ubiquitization Workflow



Adobe Technologies in Use

- Used various Adobe technologies, as a proof of concept
- These technologies are not just for the media/document/enterprise, they can be used to leverage and simplify our engineering tasks.
  - **Acrobat Test plugin**
  - **Flash**
  - **Flex**

Demo

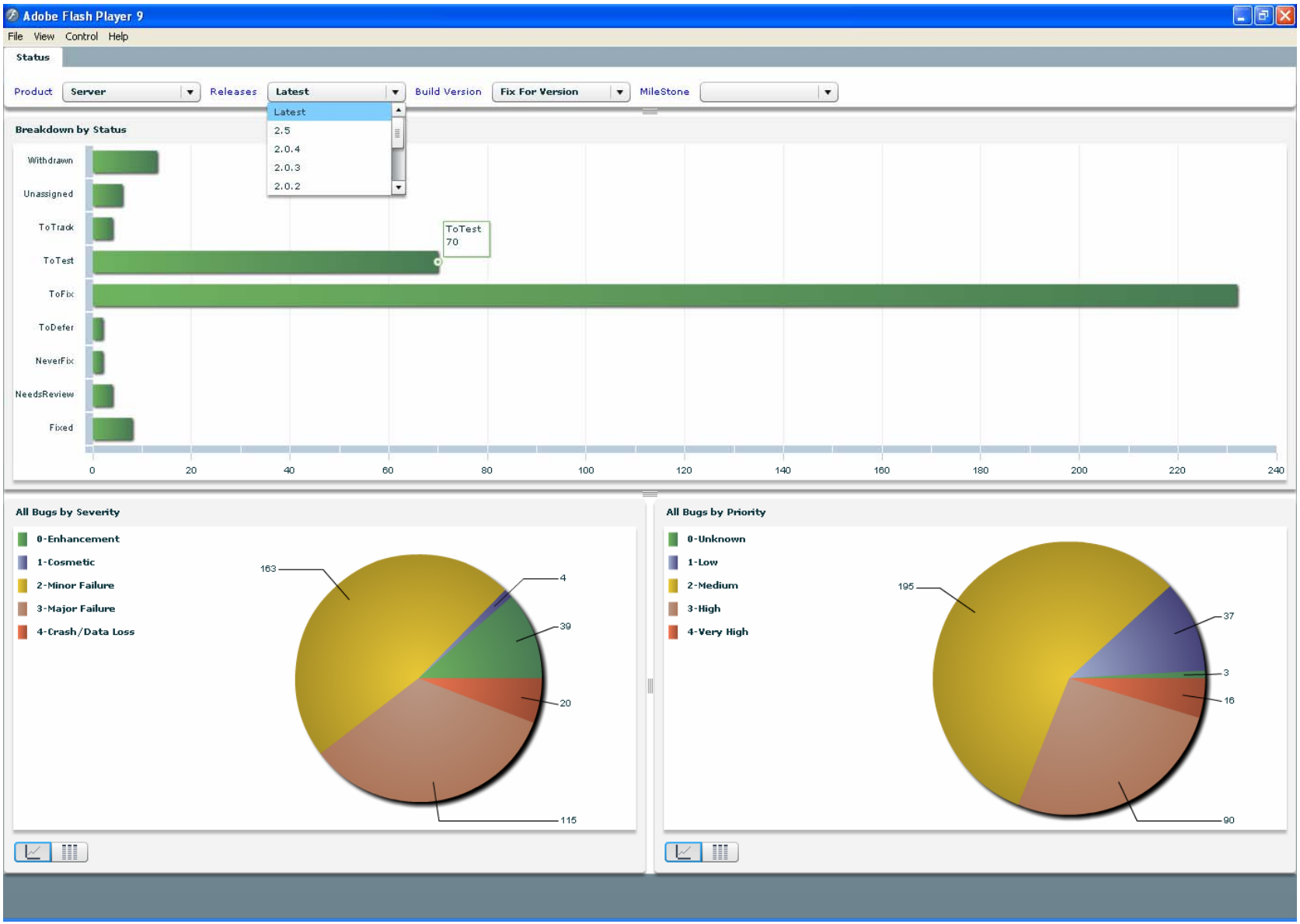
- Flash UI initiating QA workflow
- Form Fill in

Quality Center (DashBoard)

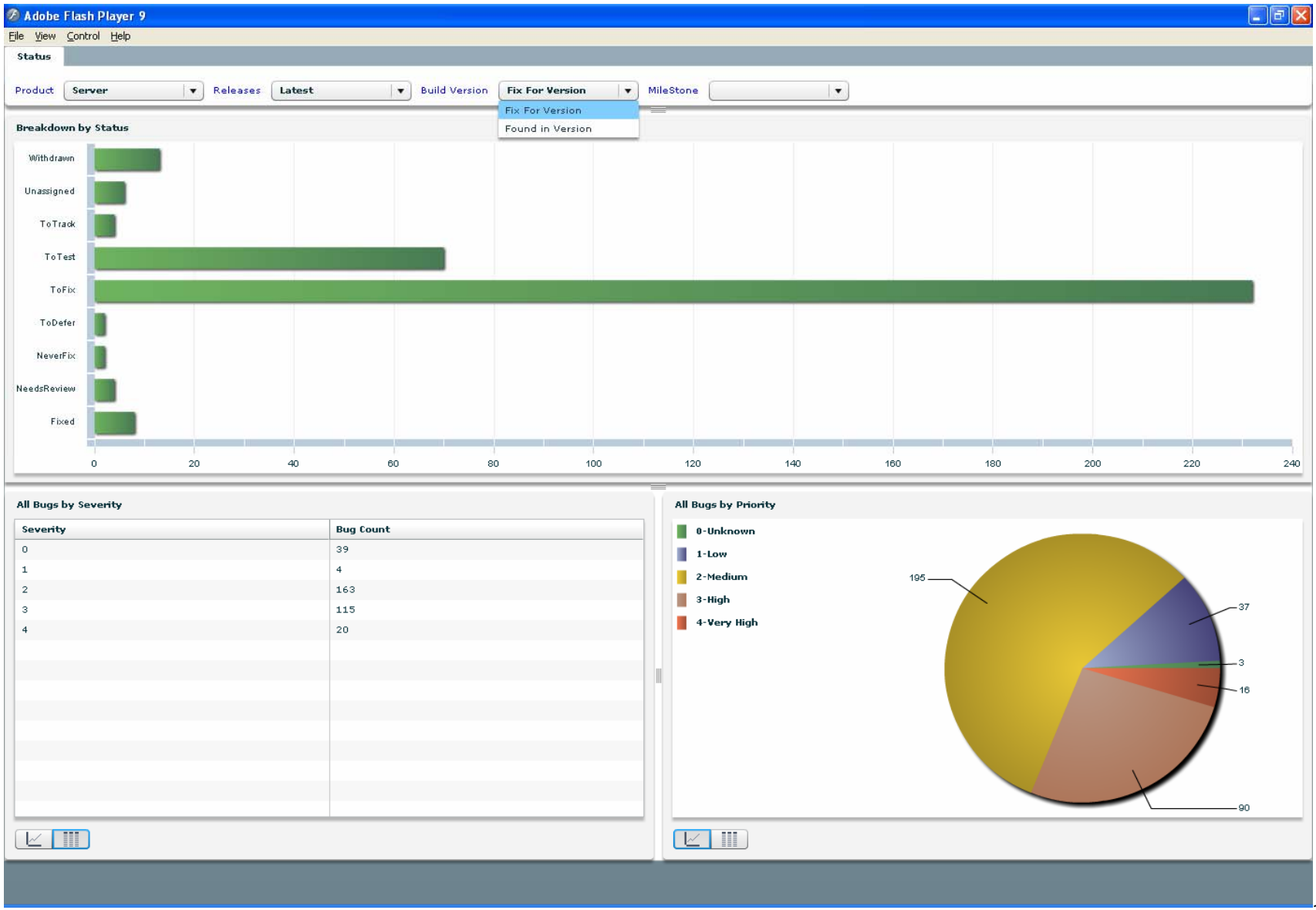
- Web based
- Real time, Rich client based product status
- Remote installation of builds and apps
- Remote execution of various test scripts
- Database driven Flex reporting
- Flex based bug analysis (through database)

Test Execution Dashboard

Bug Dashboard - 1



Bug Dashboard - 2



Followup

Questions?

Suggestions?