Assignment 3
Mini-Example of Deployment Blueprints

This document describes two example deployment blueprints created from the business blueprint in the Assignment 2 example.

Deployment Blueprint #1 (DB1) – Manual Solution

Satisfaction of domain functions by solutions *(Note: Assignment 3 also requires you to identify what components satisfy BB data and what connections satisfy I/O.)*

SB Solution Component	**BB Functions Satisfied**
Staff	• Verify enrollment • Update student profile • Verify student registration status
Student volunteers	• Process add requests • Process drop requests
Department advisors and faculty	• Approve student schedule

Allocation of solutions to deployment components

DB Component	**SB Components Allocated to DB Component**
Staff	Staff
Student volunteers	Student volunteers
Department advisors and faculty	Department advisors and faculty
Course Catalogs	N/A
Filing System	N/A

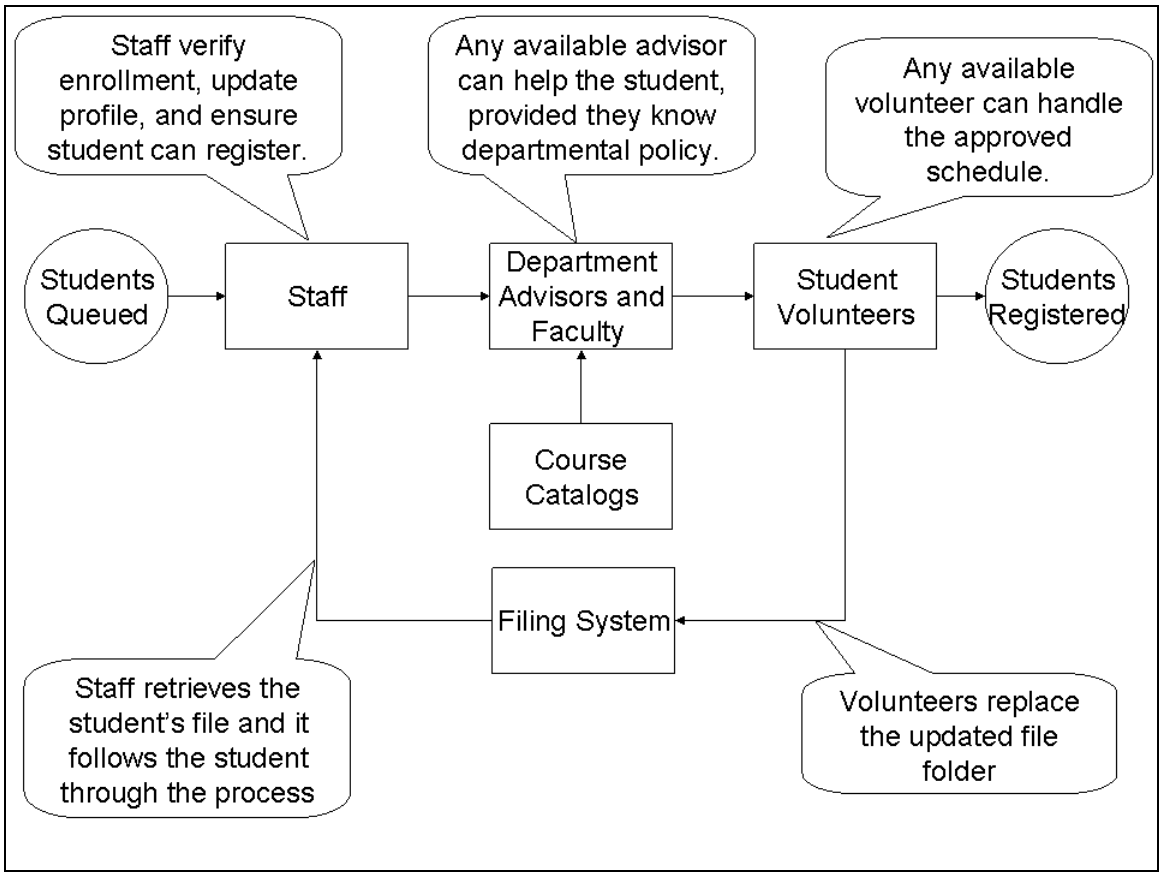

Figure 1: Graphical depiction of DB1

Rationale (Note: without full justification)

- Satisfaction of stakeholder qualities (with priorities):
    1. *Cost:* Totally manual solution (existing staff and faculty, as well as student volunteers).
    2. *Extensibility:* Can "swap out" advising experts in various disciplines.
    3. *Consistent student master record:* Student's folder is "checked out" and processed serially, ensuring only one person updating at a time; however, folder will be unavailable throughout process.
    4. *Performance:* Limited by available staff, but advising process can be broken down recursively to improve throughput.
- Design inspired by:
    - *Pipe-and-filter architecture:* Master records are "transformed" at each step in the process.
    - *Domain realities:* Deployment components closely mirror business blueprint components.

Notes

- As a totally manual implementation, the "solutions" are people playing domain roles (e.g., Staff, Student volunteers, Department advisors and faculty) and those roles also represent deployment components.

- Additional "Staff," "Student volunteers," and "Department advisors and faculty" may be used to increase throughput.
- The filing system holds folders with student records as well as class rolls.

References

- pipe-and-filter: http://www.dossier-andreas.net/software_architecture/pipe_and_filter.html

Deployment Blueprint #2 (DB2)

Satisfaction of domain functions by solutions *(Note: Assignment 3 also requires you to identify what components satisfy BB data and what connections satisfy I/O.)*

SB Solution Component	**BB Functions Satisfied**
Server authentication module	• Verify enrollment • Verify student registration status
Server profile maintenance module	• Update student profile
Server course registration module	• Process add requests • Process drop requests
Server catalog rule checkers	• Approve student schedule
Department advisors and faculty	• Approve student schedule

Allocation of solutions to deployment components

DB Component	**SB Components Allocated to DB Component**
Web Server Cluster	Server authentication module
Web Server Cluster	Server profile maintenance module
Web Server Cluster	Server course registration module
Web Server Cluster	Server catalog rule checkers (implementations of Abstract Catalog Rule Checker)
Database	N/A
Web Browser	N/A

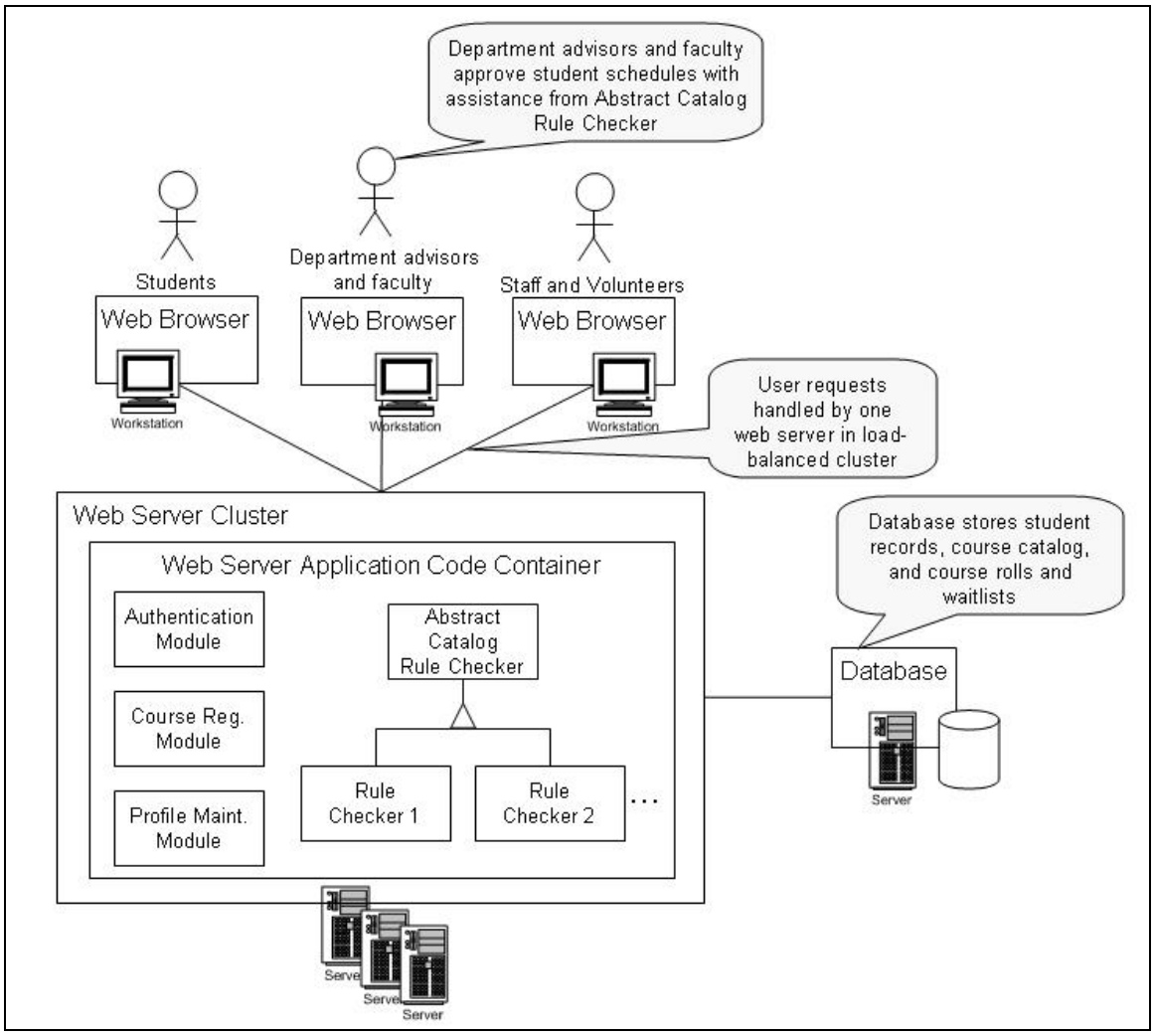

Figure 2: Graphical depiction of DB2

Rationale (Note: without full justification)

- Satisfaction of stakeholder qualities (with priorities):
  - *Performance:* Web server "farm" produces high throughput.
  - *Consistent student master record:* Student master record is locked by database, but transaction time is short, so current master record is rarely unavailable for viewing.
  - *Extensibility:* Advising staff are supported by automated rule checkers that are subclasses implementing abstract methods of the parent RuleChecker class; the subclasses can be easily swapped out or extended.
  - *Cost:* Initial hardware investment will be considerable, although much of the system-level software (e.g., web server, database) will be off-the-shelf or open source.

- Design inspired by:
    - *Client-server architecture:* Need for a central student record and course database.
    - *Object-oriented architecture:* Inheritance used for specialized rule-checkers.
    - *Web server redundancy:* Web server requests distributed across multiple servers in cluster.

References

- object-oriented architecture: http://people.cs.uchicago.edu/~mark/51050/lectures/lecture.1/lecture.1.pdf
- client-server architecture: http://en.kioskea.net/contents/cs/csintro.php3
- web server redundancy: http://www.west-wind.com/presentations/loadbalancing/networkloadbalancingwindows2003.asp

Notes

- Independent software modules run under the web server that support course registration, profile maintenance, authentication, and catalog rule checking, and these modules are "registered" to respective domain functions. However, as we discussed in class (and I mentioned in a prior e-mail), there may be multiple implementation components involved in satisfying a domain function; an "Implementation Task Decomposition" decomposes a domain function into steps representing *how* the function is implemented.
- The *Approve student schedule* function is processed jointly by *Department advisors and faculty* and *Server catalog rule checkers*.