Management and Operational Aspects of Enterprise Computing: Considerations for Curricula Development

Dan Squillace
Sr. IT Manager, Mainframe Support
R&D Data Center
SAS Institute
Cary, NC USA
Dan.Squillace@sas.com
Mission-critical Enterprise Computing systems often demand high availability and high performance for a widely-varying transaction load. These obvious requirements lead to a lot of non-obvious attributes that need to be incorporated. Many years of working in the development, management and operational sides of Enterprise Computing have generated a wish list of topics for enabling these attributes. These topics are important candidates for inclusion at appropriate points within Enterprise Computing courses. In this presentation, several such topics are discussed along with examples how they strengthen Enterprise Computing.
Topics to be covered include

- Problem determination/diagnosis infrastructure and programing conventions: Maximizing availability by shortening problem resolution time.
- Software Performance Engineering: Designing performance and scalability into applications from the beginning.
- Easing application migration through use of extensible and versioned data structures and well-defined architectural boundaries, and applying “toleration” maintenance to prior releases.
- Easing software upgrades and maintenance through strict separation of software, configuration, and data directories.
- Business Process Management: Measuring and managing applications in business-oriented terms is important for Service Level management and monitoring.
Obviously, rapid problem determination and resolution time is essential to maintaining high availability. The suggestions listed here are aimed at shortening that time. In today’s world of complex multi-system multi-layered systems, determining the cause of a problem let alone solving it is daunting.

Many times, problem determination is hindered by lack of messaging, or even worse, misleading and/or incorrect messages.

Software needs to thoroughly check return codes and other failure indicators, put out messages with sufficient information to aid diagnosis, and pass appropriate return codes up the software stack for appropriate messaging and/or error recovery at each layer in the stack. This is time-consuming but important and necessary work.

It is also useful to have a diagnostic infrastructure of the system that includes environment variables indicating the detail level of tracing and diagnostic messages and whether they should be system wide, for a specific transaction, a specific transaction instance, or a particular set of users.

There also needs to be a way to get error and diagnostic information correlated and
delivered to the staff who will analyze and respond to it. The ARM standard described at https://www.opengroup.org/management/arm/ provides a very good platform-neutral implementation for accomplishing this. While its focus is on performance, it can also be used for delivering diagnostic information.
Many large applications systems have suffered preventable severe performance problems. Two of the root causes are the mind set and the training analysts and programmers bring to the job. They are well-versed in languages, algorithms, and small systems but have little understanding of the dynamic nature of resource consumption in large systems. I liken that to looking at a static road map rather than seeing a dynamic view of traffic flow.

Furthermore, any measurement tends to focus on elapsed time, not resource consumption.

Students need to gain an intuitive feel as well as a methodology for determining which software paths are critical because of their frequency of use and/or path length.

Incorporating SPE into CS and IT curricula would go a long way toward meeting these needs.
All of these items can improve availability by enabling application upgrade migration through operation of multiple releases simultaneously and a permitting business-driven phased migration to new releases.

Fallback, when needed, is also easier because the previous release of the system should be able to operate using the data structures and ignoring elements not applicable to the older release.

For two good examples of extensible data structures that can be processed by multiple software levels, see the IBM z/OS Type 30 SMF and Type 70 RMF record descriptions.
Strict separation of these materials makes implementation of and back out from software changes much easier. Software changes can be rolled in and out without perturbing the site-specific configuration information and business data.
Successful management of enterprise class applications requires that such management use terminology and measurements appropriate to the business function being supported. Such measures resonate with business managers and users and are consistent with their view of applications. Furthermore, they are essential to addressing problems such as the following.

Network traffic is high and hence response time is poor. There are no identifiable network failures. So is the problem due to:
- Increased transaction volume?
- A significant change in transaction mix?
- Application changes resulting in increased utilization per transaction?
- Bringing an additional group of users online?
Response Time Quality of Service is a more accurate measure of business work being delivered and the productivity of employees (or customers!) using the application.

It is useful to look at response time in terms of business process and sub-processes as well as transactions. Traditionally, IT has focused on the response time of individual transactions presented to the system. The time between transactions, called “think time” was of no concern other than in how it affects transaction rate.

However, it is more appropriate to view that time instead as the manual portion of the process which is as subject to problems and need for improvements as the computerized portion of the process. Confusing panels, incomplete instructions, and unclear input field requests are examples.

Even environmental issues such as faded LCD panels or sunshine at certain times of the day can affect the manual portion of the business process.
Response Time Tied to an Operational Business Process which is ....

.... The tasks, both manual and automated, required to deliver a service to a customer
This chart illustrates the concepts of a business process, sub processes, and transactions. The business process in this case is “Customer Order” with two sub-processes, “Item Selection” and “Order Fulfillment”. There may be multiple iterations on “Item Selection”.

And, of course, “Item Selection” and “order Fulfillment” may have additional sub-processes of their own.
Specific Student Activity and Topic Suggestions

- Construct system analysis and programming group exercises that focus on compatibly extending a previous system.
- Incorporate basic efficiency techniques into programming exercises.
- Incorporate requirements for error checking and appropriate error messages.
- Whenever possible incorporate reinforcement of written and oral communication skills and critical thinking into class exercises.
“A problem well-put is half solved.”

John Dewey quoted in “The Chronicle of Higher Education”

This quote, which is in the June 17th 2011 issue of “The Week” magazine struck me as quite appropriate this talk. Based on my experience it rings very true.

On a related note, as an R&D Product Development Manager, during team meetings when folks proposed a change or enhancement, I asked them to first state clearly what the problem is they were trying to solve.