: What are common DSS architectural patterns or styles?

by Dan Power

Editor, DSSResources.com

Information technology continues to evolve, but one can identify various architectural patterns or styles
that seem to reoccur in the development of computerized decision support systems. New systems seem to
imitate or adopt prior patterns by incorporating updated hardware and networking technologies. It is likely
historical architectural patterns will persist with service oriented and message-based implementations
of decision support systems (cf., Natis, 2003; Whetten, 2001).

Advocates of a unified modeling approach to building systems perceive that design patterns or common
architectural styles exist for classes of systems with similar purposes (cf., Booch, Rumbaugh and Jacobson, 1999;
Eeles, 2006; Gamma, Helm, Johnson and Vlissides, 1995). Inadequate attention has been given to defining these
patterns or styles for DSS, but some material that has been written on the topic.

An architecture for a computerized Decision Support System documents the plan for deploying the components of
the envisioned DSS or how the components were actually deployed in an implemented decision support application. In
general, DSS architecture specifications focus on the dialog/user interface, model base and data base components
and how they are interconnected. "Architecture is the fundamental organization of a system
embodied in its components, their relationships to each other, and to the environment, and
the principles guiding its design and evolution (IEEE 1471-2000)."

According to Sprague and Carlson (1982), the components of the DSS technology framework include
dialogue management, data base management, model base management, and DSS architecture. They
argued the DSS architecture describes the mechanism and structure for the integration of
the dialogue, data base and model management components. They identified 4 architectures:
the DSS Network, the DSS Bridge, the DSS Sandwich, and the DSS Tower. The DSS Network has

multiple dialogue, modelling and data base components that are interconnected and can share data through a component interface. The Bridge has a standard interface with local dialogue and modeling components that link to remote modelling and data base components. A Sandwich architecture has a single dialogue and data base component, but multiple model components are linked by the architecture. The dialogue and data base components are the "bread" and the model components provide the "meat" for the application. Finally, the Tower includes more vertical components or tiers with data extraction tools integrating diverse data base components. The rest of the Tower architecture is similar to a Network structure.

Power and Kaparthi (1998) identified six DSS architectures: distributed dialogue, remote dialogue, distributed model, distributed data, remote data and stand alone. The distributed dialogue is basically a thin-client web architecture with the dialogue presented on the client and with models and data accessed from one or more servers using a network connection. The remote dialog is a more traditional thick-client application with the entire dialogue interface on the client and the model and data base components on one of more servers. In the distributed model, the application software on the client expands and model capabilities are distributed for more efficient processing. The distributed data architecture requires accessing data across the network for processing. With a remote data archictecture, some data is download to the client for faster processing. Finally, a stand alone architecture has the entire DSS application on a stand alone computer with no provision for network access to server based components.

In a similar framework, Schay (1992) of the Gartner Group defined five different styles of client-server computing. The difference in these styles was the portion of the computing process that is "distributed" to another computer over the network. The five styles are defined in relation to the three main processes in a computerized application: (1) presentation (user interface), (2) process (application logic) and (3) data storage (data management). The five styles are called: 1) Distributed presentation, 2) Remote presentation, 3) Distributed logic, 4) Remote data management, and 5) Distributed database. The descriptive names capture the architectural differences.

Most architectural patterns or styles are very general. According to Eeles (2006), "An important aspect of an architecture is not just the end result, the architecture itself, but the rationale for why it is the way it is. Thus, an important consideration is to ensure that you document the decisions that have led to this architecture and the rationale for those decisions." Potentially generic architecture patterns can lead to a better understanding of how to build computerized decision support systems. DSS of

all types, communications-driven, data-driven, document-driven, knowledge-driven and model-driven,
are increasingly integrated with other Information Systems and a given DSS may have multiple decision
support subsystems of different types. The architectures of data-driven DSS emphasizes database performance
and scalability. Most model-driven DSS architectures store the model software on a server and distribute the user
interface software to clients. Networking issues create challenges for many types of DSS but especially
for a geographically distributed, communications-driven DSS. Much more needs to be done to model the
increasingly complex patterns in DSS.

Historically the focus has been on structural software system components of DSS, but there is an increasing need
to identify patterns in other "views" for particular types of DSS. "Views" are analogous
to the different blueprints created for a complex building by an architect.
A DSS architecture can potentially be diagrammed in terms of four layers: the business process map, the systems
architecture, the technical architecture, and an output delivery architecture. The business process map shows how decision making tasks are completed. The systems architecture shows the traditional software components. The
technical architecture focuses on computing hardware, protocols and networking. The output delivery architecture
focuses on the results and representations of the system (cf., Power, 2002). Identifying patterns in process maps
and output delivery can also assist DSS architects.

Having a well-defined and well-communicated architecture for a specific DSS provides an organization
with significant benefits. An architecture diagram helps developers work together, improves planning, increases
the development team's ability to communicate system concepts to management, increases the team's ability
to communicate needs to potential vendors, and increases the ability of other groups to implement systems
that must work with the specific DSS. Technical benefits of defining a DSS architecture include the ability to plan systems
in an effective and coordinated fashion and to evaluate technology options within a context of how they
will work rather than abstractly. A specific DSS vision and an architecture for a new DSS helps communicate
the future and provides a consistent goal for making individual design decisions. Achieving all these benefits

requires that both information system professionals and prospective DSS users must cooperate closely in defining the
intended architecture. Design patterns can help DSS architects and potential users evaluate and select
solutions. Identifying patterns or styles can also assist a DSS architect in preparing early-phase project
estimates to make a business case for a proposed decision support system.

Design the DSS before you build it. As always, your comments, suggestions and feedback are welcomed.

References

Blythe, K. C., "Client-server Computing Management Issues," CAUSE/EFFECT, Volume 16, Number 2, Summer
1993, URL http://www.educause.edu/ir/library/text/CEM9320.txt .

Booch, G., J. Rumbaugh, and I. Jacobson, The Unified Modeling Language User Guide, Boston: Addison-Wesley, 1999.

Eeles, P., What is a software architecture? IBM, February 15, 2006,
URL http://www-128.ibm.com/developerworks/rational/library/feb06/eeles/ .

Gamma, E., R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements
of Reusable Object-Oriented Software, Boston: Addison-Wesley, 1995.

IEEE Std 1471-2000, "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems,"
URL http://standards.ieee.org/reading/ieee/std_public/description/se/1471-2000_desc.html .

Mallach, E. G. Understanding Decision Support and Expert Systems. Burr Ridge, IL: Richard D. Irwin, Inc., 1994.

Natis, Y. V., "Service-Oriented Architecture Scenario," Gartner, April 16, 2003, URL http://www.gartner.com/DisplayDocument?doc_cd=114358 .

Nolan, R. L. "Building the company's computer architecture strategic plan." Stage by Stage (Nolan, Norton & Company) 2 (Winter): 1983: 1-7.

Power, D. J. and S. Kaparthi. "The Changing Technological Context of Decision Support Systems", In Berkeley, D., G. Widmeyer, P. Brezillion & V. Rajkovic (Eds.) Context-Sensitive Decision Support Systems. London: Chapman and Hall, 1998.

Schay, P., "How will Traditional Mainframe and Midrange Systems Evolve to Support Client/Server Computing?" in
the Proceedings of the Gartner Group Symposium 1992, Scenarios, Vol. 1 (Stamford, Conn.: Gartner Group, 1992), p. 6 of Client/Server section.

Sprague, R.H. and E.D. Carlson. Building Effective Decision Support Systems. Englewood Cliffs, NJ: Prentice-Hall, 1982.

Welsh, M. J., "Enterprise architecture essentials, Part 3: Design and build your enterprise architecture," IBM, September 11, 2007, URL http://www.ibm.com/developerworks/library/ar-enterarch3/ .

Whetten, B., "Message-Based Computing: The Fourth Wave of Integration," 12/23/2001, URL http://www.ebizq.net/topics/jms/features/1579.html?&pp=1 .

Author: Daniel Power
Last update: 2007-10-03 02:04