Characterization Framework and Design Patterns for the Disadvantaged User

Dr. Fatma Dandashi, Aaron Griggs, Jeffrey Higginson, James Hughes, Wilson Narvaez. Dr. Marwan Sabbouh, Salim Semy, Dr. Beth Yost
The MITRE Corporation
Dandashi, agriggs, higginso, jdhughes, wnarvaez, ms, ssemy, bethyost@mitre.org

Abstract

Many Service Oriented Architecture (SOA) approaches in use today presume the consistent availability of reliable networks and limitless resources. For some Department of Defense (DoD) and other Government users, operating at the tactical edge, who may be disadvantaged in terms of network and resource availability, the current methods of development may not provide them with reliable capability. In this paper, we propose a method for capturing design patterns for the disadvantaged user using the common vocabulary of a characterization framework. We also provide a set of design patterns that minimize technical constraints, and derive the infrastructure requirements needed to implement a selected design pattern.

1. Introduction

Many Service Oriented Architecture (SOA) approaches in use today presume the consistent availability of reliable networks and limitless resources. This is often not the case for a Department of Defense (DoD) or other Government user operating at the tactical edge who may be disadvantaged in terms of network and resource availability. For such a user (e.g., a soldier at the battlefield, or an emergency first responder), current methods of SOA development may not provide reliable capability. The challenge then lies in determining how to implement service-based solutions for the disadvantaged user.

In this paper, we propose a method for addressing SOA implementations within the tactical edge environment. To do this, we propose a characterization framework and common vocabulary to describe various environments, and capture design patterns using the common vocabulary. We apply this characterization framework to a particular use case, identifying a set of design patterns that minimize technical constraints associated with the use case. This application serves as a reference implementation for the characterization framework. Finally, we highlight how design patterns can be used to derive infrastructure requirements, and conclude with a brief discussion of the implications of this work, its applicability to any disadvantaged user, and describe possible future work.

2. Characterization Framework and Common Vocabulary

The Characterization Framework is aimed at defining the technical constraints in providing service-based capabilities to a disadvantaged user. To define the framework and help identify design patterns, we began by gathering use cases, identifying common characteristics of various environments, and then focusing on defining a common vocabulary to describe those environments. Four environments were identified: fixed center, mobile center, mobile swarm, and dismounted. Each environment was then characterized by four dimensions:

1. The availability and robustness of a network
2. The availability of resources to execute a particular function
3. Information Assurance (IA)
4. User Interface (UI).

These four dimensions were further quantified using a set of attributes and a range of possible values for each attribute. The network dimension was characterized by the attributes: connectivity, bandwidth, and latency, where both latency and bandwidth (i.e., speed and capacity) of the network define the throughput of the network. The resource dimension was characterized by the attributes: processing capacity, storage capacity, power, total system space, and total system weight. The IA dimension was characterized by the attributes: fixed network topologies, network defenses, host defenses,

perimeter defenses, policies & procedures, and data defenses. The last dimension, UI, was characterized by the attributes: content, standard user interface, system training, receptiveness, decision time, lighting, environment, display, output, and input.

Figure 1 illustrates how the four environments of fixed center, mobile center, mobile swarm, and dismounted were characterized for each dimension. Values defined for each attribute appear in cells as detailed below. The classes shown in Figure 1 serve as the representational set of environments for which design patterns can be specified. There is nothing DoD unique about these environments or their attributes as defined by this characterization framework. It is therefore possible to apply and use the framework and the common vocabulary to address the needs of non-governmental users.

3. Design Patterns

Design patterns occur in many different disciplines. The concept of design patterns is summarized by the architect Christopher Alexander as a manner to "Describe a problem which occurs over and over again in our environment, and then describe the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice." [1] The computer science discipline later adopted Alexander's idea and summarized design patterns as "a description of communicating objects and classes that are customized to solve a general design problem in a particular context." [2]

3.1 Design Pattern Definition Template

Each design pattern was defined using a standard template. This template was intended to provide the minimally complete set of information necessary to support the use of the design pattern by information services developers (and the supporting test and evaluation communities, users, etc.). As defined here, each pattern starts with a unique name and description, followed by the context and problem statement, an application example, a description of the pattern's trade-offs, and an illustration. We note that no document could contain a complete set of possible design patterns, and any such attempt would certainly end in failure (after an exhausting search effort). Rather, this paper contains a representative set of patterns for each of the main dimensions of the environments and a template that can be used to capture, enable reuse, and refine evolving patterns. Over time, some of the patterns we recognize as useful today may become unnecessary or obsolete, replaced by more efficient approaches enabled by advancing technology, infrastructure improvements, or changes in operational constraints.

Some patterns may be most useful when applied in composition with other patterns, in order to provide an adequate solution for a particular problem. In such cases, the problem definition is divided into a set of sub-problems and specific design patterns are defined, where applicable, for each sub-problem. In this manner, the composition of the identified patterns provides the solution to the overall problem.

3.2 Types of Design Patterns

This section lists example design patterns that are categorized into four sets, based on their use.

1. The first set consists of patterns that alleviate resource and network constraints:
   - Messaging Bridge (MB) [3]: The data source service sends the message to the MB Source and the MB Source performs the actual transmission. Similarly, there is an MB Destination that receives a message and delivers it to the destination.
   - Notification [4]: Availability of new content is broadcast to all interested consumers that subscribed to it.
   - Personalized Delivery [4]: Intermediate service provides a customized data and interface to the end service requestor, based on a user profile.
2. The second set consists of patterns that alleviate resource constraints:
   - Reliable Asynchronous Messaging [4]: Messages produced by the service provider in response to the request are queued until the service requestor asks for the messages.
   - Store and Forward [5]: Network of nodes receive data, store data until connectivity is re-established, then forward data to other nodes.
   - Caching [6]: Replicate and synchronize data within local data stores to facilitate data retrieval.
   - Compression [7]: Compress the data for optimal use of bandwidth during transmission.
   - Publish and Subscribe [4]: Data consumers register subscriptions. When the data is available, it is automatically published by data providers to consumers.

3. The third set consists of patterns used for IA purposes. The following example design patterns have been excerpted from [8]
   - Simple Firewall Configuration: A firewall inspects and filters incoming and outgoing network traffic based on the protocol, port number, and the type of application service to be accessed or type of application service requesting access.
   - Demilitarized Zone: A Demilitarized Zone (DMZ) permits different protection roles to systems on the DMZ than internal systems. Typically, systems on the DMZ require less protection than internal systems, as they can be accessed from the World Wide Web.
   - Multilevel Security: In some environments, data and documents may have critical value and their disclosure could result in serious problems. This pattern describes how to categorize sensitive information and prevent its disclosure. It discusses how to assign classifications (clearances) to users and classifications (sensitivity levels) to data, and how to separate different organizational units into categories. Access of users to data is based on policies, while changes to the classifications are performed by trusted processes that are allowed to violate the policies.
4. The fourth set consists of patterns for the design of the UI. Examples are:
   - Canned Messages**:** Users can choose from a list of predetermined messages, rather than having to enter text.
   - Flattened Navigation: Users can select an option with a single click, rather than navigating through a series of cascading menus.

4. Reference Implementation

There is a long list of design patterns available in the literature [2,3,9,10,11]. However, our challenge was to identify particular design patterns that alleviate the technical constraints associated with the disadvantaged user. The purpose is to demonstrate the use of the common vocabulary to describe design patterns for a particular use case. In this section, we apply the identified design patterns to a use case example and describe a composite design pattern in terms of its name and description, context, problem, proposed solution, Trade-offs, and some illustrations of the use case solution.

4.1 Data Dissemination

Name: Data Dissemination (Pattern Composition)

Description: Move messages and data across environments (i.e., from the mobile center to the mobile swarm to the dismounted environments).

Use Case Description: A Tactical Operations Center (TOC), classified as a Mobile Center environment, sends various messages, e.g., Operations Orders to a multimedia High Mobility Multipurpose Wheeled Vehicle (HMMW-V), also known as a humvee or hummer, base station. The HMMW-V, classified as a mobile swarm environment, is employed to support dismounted squad operations using heterogeneous (multiple disparate) communications media. The HMMW-V acts as a Combat Vehicle Heterogeneous Cell site (CVHC). Dismounts are typically equipped with small commercial off-the-shelf handheld radios and readily available devices that enable their operations to be conducted unconstrained by utilizing the CVHC parked in their vicinity. The important aspect of this is the assumption that dismounts typically operate within line-of-site communications range of their supporting CVHC for most operations, which enables en-route coordination and planning. In these circumstances, dismounts may partition from the global network connecting the TOC to the CVHC, while remaining connected to the vehicle on another local network. Dismounts can still be provided with messages and order updates delivered directly to their handhelds. Such order updates may include a picture of a known terrorist or his last reported location. Similarly, a dismount may send a picture of a detained person up to the vehicle, which in turn may relay it to the TOC for identification against a watch list database.

Context: Mobile Center – Mobile Swarm – Dismounted

Problem:

- **Connectivity:** Well Connected – Intermittently Connected – Mostly Disconnected
- **Bandwidth:** High – Medium/Low
- **Processing Power:** Servers/Multiple Workstations – Single Workstation/ Handhelds
- **Storage:** Large Data Storage Devices – Single Hard Drives/Memory
- **Display:** Multiple Displays – Single Display

Application Example: A Data Dissemination Service (DDS) implementing a Publish and Subscribe mechanism is employed with the Messaging Bridge design pattern to alleviate problems associated with the tactical edge.

Trade-offs:

- **Benefit**s: DDS uses standard Web protocols to transfer the content. As such, DDS requires high throughput networks and full connectivity for its operations. Since that is not available for the use case described above, we identified and employed the Messaging Bridge design pattern to alleviate the technical constraints imposed at the tactical edge. The Messaging Bridge pattern shields the DDS from intermittent connectivity and low bandwidth associated with the tactical edge. The Messaging Bridge pattern accomplishes this by implementing connection pooling. The Messaging Bridge pattern also implements queuing and compression to address intermittent connectivity and low bandwidth. The Messaging Bridge pattern can also cache data, provide a specified quality of service, and more optimally continue with sending a message from the point of the network disruption in the case of a large message.
- **Limitations**: The addition of the Messaging Bridge to the DDS results in the significant increase of the DDS footprint, making it more difficult to deploy DDS in storage and processing challenged environments.

Illustration: Figure 2 illustrates the DDS with Messaging Bridge architecture. In a typical publish and subscribe paradigm, content providers publish their content to the data dissemination server, while consumers receive certain content by first registering their interest with the data dissemination server. With the insertion of the Messaging Bridge, the DDS server disseminates content through the DDS client proxy, which is typically co-located with it (the server). Similarly, the DDS client registers its interest in a particular content through the DDS server proxy, which is co-located with it (the client). Both DDS server and client Messaging Bridge are implemented as Mule [12] servers. The Mule Enterprise Service Bus (ESB) provides support for connection reestablishment after a dropped connection, and facilities for compression and queuing.

4.2 Infrastructure Requirements for Design Patterns

For the use case described above, a number of infrastructure components are required to implement the solution. These components include an information exchange infrastructure to work across constrained networks, as well as client-side applications supporting offline mode.

To handle network disruptions, a proxy service is an important infrastructure component. In our solution, the proxy's implementation is facilitated by the open-source Mule ESB [12]. This implementation is well suited for message-oriented information exchange. The Mule ESB also provides a platform to integrate additional patterns, such as compression. In some cases, dealing effectively with network disruptions at the tactical edge cannot be solved at the application layer alone, but requires architectures spanning the messaging layer, middleware layer, application server, and browser. An example of such an architecture is the Disruption Tolerant Network [13] being developed at the Defense Advanced Research Projects Agency (DARPA), which makes use of store and forward techniques, routing models, and persistence to overcome disruption in a network.

On the client side, supporting an offline mode for client applications is an important aspect in dealing with intermittent networks. Lately, we have witnessed new developments concerning the interactions between application servers and browsers. In particular, the Google Gears Toolkit and Dojo allow Web developers to program applications to support an offline mode of operations in addition to the online mode. These solutions work by implementing a local server on the browser side where resources are cached. Updates from the server are retrieved when resources are requested. Follow on work may include testing and incorporating these new solutions.

5. Conclusion

In this paper, we proposed a method for capturing design patterns for the tactical edge using the common vocabulary of the characterization framework. We provided a set of design patterns that minimize technical constraints associated with the disadvantaged user and derived the infrastructure requirements needed to implement a design pattern. This implementation serves as a reference, geared toward demonstrating the use of the characterization framework and validating the design patterns.

The characterization framework and the use of design patterns as proposed in this paper provides a number of benefits:

- The common vocabulary can be used to describe various environments such that one can look across multiple use cases and identify commonality in the type of constraints introduced in each use case. Subsequently, this allows for sharing of design patterns and implementation solutions across these use cases.

- The framework provides the basis for a process to assess the readiness of a particular existing system for the tactical edge, comparing the implementation against appropriate design patterns and infrastructure requirements for particular classes of environments.
- The use of design patterns particular to disadvantaged environments provides an ability to make investment decisions on infrastructure requirements as part of infrastructure upgrades and resource prioritizations.

Next steps for the characterization framework include applying the framework to additional use cases to validate the classes of environments and associated attributes. Follow-on activities may focus on identifying non-DoD use cases to validate the adoption of this approach for non-disadvantaged users as well.

6. References

[1] Alexander, C., S. Ishikawa, M. Silverstein, 1977, "A Pattern Language: Towns/Buildings/Construction," New York: Oxford University Press. ISBN 0-19-501919-9.

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

[3] Hohpe, G., B. Woolfe, 2004, "Enterprise Integration Patterns, Designing, Building, and Deploying Messaging Solutions," Boston, MA: Addison-Wesley.

[4] "ws-Notification, ws-pubsub, & personalized delivery standards," Contributors: IBM, Akamai Technologies, Computer Associates International, SAP AG, Fujitsu Laboratories of Europe, Globus, Hewlett-Packard, Sonic Software, TIBCO Software, http://www.ibm.com/developerworks/library/specification/ws-notification/, http://www.ibm.com/developerworks/patterns/portal/access-personalized-runtime.html, http://www.ibm.com/developerworks/library/specification/ws-rm/, http://www.ibm.com/developerworks/library/specification/ws-pubsub/,

[5] Chappell, D., 2004, "Enterprise Service Bus," Cambridge, MA: O'Reilly.

[6] Srinivasan, H., J. Conallen, E. Lane, "Building SOA Applications With Reusable Assets, Part 4: The Requester-Side Caching Pattern," http://www-128.ibm.com/developerworks/library/ws-soa-reuse4/index.html

[7] ISO/IEC 24824-1, "Information Technology -- Generic applications of ASN.1: Fast Infoset," March 30, 2007, http://www.iso.org/ Schumacher, M. et al., 2005, "Security Patterns: Integrating Security and Systems Engineering," Indianapolis, IN: John Wiley & Sons.

[8] Schumacher, M. et al., 2005, "Security Patterns: Integrating Security and Systems Engineering," Indianapolis, IN: John Wiley & Sons.

[9] Adams, J., S. Koushik, G. Vasudeva, G. Galambos, 2004, "Patterns for e-business, A Strategy for Reuse," IBM Corporation, http://www-128.ibm.com/developerworks/patterns/

[10] Endrei, M., et al., 2004, "Patterns: Service-Oriented Architectures and Web Services, IBM RedBooks, IBM. http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246303.html?Open

[11] Fowler, M., August 2006, "Writing Software Patterns," http://www.martinfowler.com/articles/writingPatterns.html

[12] Open Source Mule ESB (Enterprise Service Bus) http://mule.codehaus.org/display/MULE/Home

[13] DARPA, "Disruption Tolerant Network," http://www.mitre.org/news/events/tech05/briefings/2184.pdf.

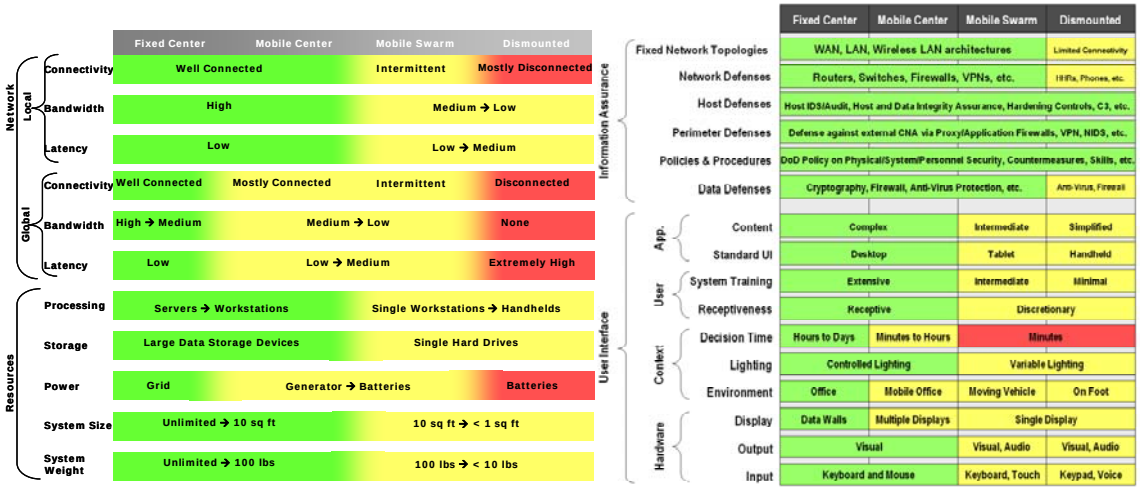

Figure 1. Summary of Environments' Characterization

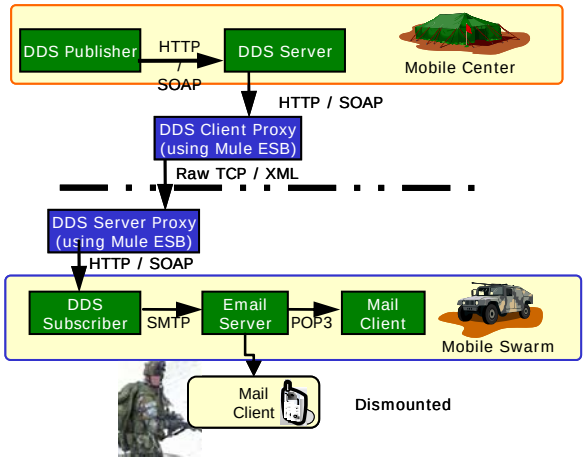

Figure 2. Illustration of DDS with Messaging Bridge