A Dynamic Menu Interface Designer

Copyright © 2011 by Elvis C. Foster, Jesse E. Schmidt, and James Dahlen

Elvis C. Foster, PhD
Assistant Professor of Computer Science
Department of Computer Science,
Keene State College,
Keene, NH 03435
efoster@keene.edu; elfoss01@gmail.com

Presented at the ATINER Conference for Computer Science in Athens, June 2011

Published in Papadopoulos & Petratos; *Enterprise Management Information Systems;* University of Hull, California State University Stanislaus, 2012

Elvis C. Foster is Assistant Professor of Computer Science at Keene State College, New Hampshire. He holds a PhD in Computer Science from University of the West Indies, Mona, Jamaica, and is a PhD student of Organizational Management at Capella University, Minneapolis, Minnesota. Among his varied interests is a focus on how more effective strategic information systems can be developed for the business environment.

Jesse E. Schmidt is a former student of Keene State College. He is currently pursuing his MS in Computer Science at University of New Hampshire.

James J. Dahlen is a former student of Keene State College. He now operates as Mobile Application Developer at New Wave Industries in Newington, Connecticut.

Abstract

Contemporary software engineering is typically influenced by critical success factors including development speed, precision, interoperability, user-friendliness, usefulness, and reusability. Software consumers have become quite impatient, and reluctant about persisting with software systems that do not meet their expectations. Moreover, software developers are expected to deliver projects on or ahead of schedule, or face the wrath of disgruntled consumer(s).

This paper proposes a *dynamic menu interface designer* (DMID) as a software component that has the potential of reducing software development duration. The DMID takes as input a data set that includes the essential information on the operational and security requirements of the software system being constructed, and generates a menu of user options based on each user's profile. This component removes the burden of menu design and construction from the software construction phase of the *software development life cycle* (SDLC), thus giving the software engineer more time to concentrate on other pressing and important aspects of software construction.

The paper proceeds under five sections: Section 1 underscores the importance of good user interface design as an important component of software design. It ends by introducing the idea of a DMID. Section 2 provides a rationale for the DMID, showing how it could significantly reduce the development time for a software engineering project. Next, the architecture of the DMID is discussed in section 3. This is done from a database perspective, as well as a user interface perspective. Section 4 briefly describes a prototype of the DMID that has been developed and tested. Finally, section 5 provides a summary and some concluding remarks.

Keywords: Software Construction; Interoperability; Ubiquitous Software Component; Software Documentation; User Interface Design.

1. Introduction: Importance of Software User Interface

Anyone who has been intricately involved in the engineering and construction of computer software for the business environment will readily admit that user interface design and construction is a very important aspect of software development. It is also well known that planning, constructing, and testing the user interface for a software system takes considerable time and effort. In fact, a research done at Oulu University shows that in a particular project that was studied, user interface design took relative more time than other aspects of the software design (see [Kivisto, 2000]). This is by no means alarming. A user interface is the window through which end-users access the software system (see [Foster, 2010]); therefore, taking meticulous steps to get it right is of paramount importance.

One urgent concern in the software industry today is to create more complex software at a faster pace and at lower costs. The industry demands at reduced cost, quantum leaps in complexity, reliability, design capability, flexibility, speed of development, ease of change, ease of usage, accuracy, interoperability, user friendliness, and usefulness (see [Martin, 1993]). Moreover, with increased technological capabilities, the software consuming public has become much more demanding about computer software, and less forgiving when requirements are not met. Indeed, we are seeing in our time, a gradual decrease in the level of tolerance for runaway projects [Glass, 1998].

How do we further improve on productivity in the discipline of software construction? We know that it is quite a technical and knowledge intensive discipline [Robillard, 1999], but we also know that it is very competitive and dynamic. One way to face this challenge is perhaps to reduce the development time on a project. A *dynamic menu interface designer (DMID)* is a software component with the potential of assisting in the achievement this objective.

2. Rationale for a Dynamic Menu Interface Designer

Research such as [Chan, 1998], [Curl, 1998] and [Khalifa, 1998] emphasize the importance of user interface design and development during software construction. Indeed, authors such as [Shneiderman, 2005] and [Nielsen, 1993, 1999] not only underscore the importance of this aspect of software engineering, but provide us with ample guidelines and principles for good user interface design. Moreover, we know that the user interface is in many cases, the end-user's (only) perspective of the software. Having a carefully planned user interface is therefore of paramount importance, since it affects user acceptance and by extension, the success of the software.

Even with much experience, designing and constructing the user interface takes time (and if you employ user interface engineers, that means additional expense). Furthermore, there is a close nexus between user interface and system security, which must not be missed during software construction: Users use the user interface to gain access to the resources of the system; accessibility to system resources implies system security. In many business applications and software systems, evidence of appreciation of the importance of this nexus is missing. For instance, it should be possible to access a given system option from different menus. It should also be possible to tie in authentication features into the menu interface, in a seamless, transparent manner. Yet, in many products, these two basic requirements are either unacceptably provided, or completely missing. But all the major CASE tools, RDBMSs and RAD tools provide the facility for user interface design, so why the fuss? This leads us to the third problem:

for most of these software development environments, the facility for user interface design is intricately associated with the software product itself, so that having the facility means acquiring the software product. Two classic examples of this scenario are the Authorization Manager for the Windows Server environment (see [Microsoft, 2010a] and [Microsoft, 2010b]), and the Windows Presentation Foundation (WPF) for the .NET framework (see [Microsoft, 2010c]). The former facilitates easy role-based administration of user privileges; the latter facilitates the creation and management of a wide range of stand-alone as well as browser-hosted applications.

Suppose that we had a generic software component that could be used on any software engineering project, in respect of user accessibility to system options. Suppose further, that the user interface provided by this software, was a user friendly graphic user interface (GUI), following required principles and guidelines (for instance [Shneiderman, 2005] and [Nielsen, 1993]). This software system accepts as input, essential details for the targeted software system, user options, menu details, and user authorization requirements. At sign-on, it builds dynamically, the user's menu, based on the user authorization log and other related information, stored in an internal database. Let us give this software a name: a *Dynamic Menu Interface Designer* (DMID). This concept has been successfully implemented in Web-based systems such as AntsSoft's Ultra Menu (see [AntsSoft, 2009]) and Drupal (see [Drupal, 2010]). However, in the areas of business applications and information systems (IS), there is a need for such a product.

The DMID has as its objective, platform independence, and has the potential of bringing a number of advantages to the software engineering experience, particularly in the area of information systems (IS) development. Included among the proposed advantages are the following:

- User interface specification is significantly simplified by transforming the problem to mere data entry. By providing the facilities for menus to be defined (via data entry) and loaded dynamically, based on the user's access rights, the software engineer is spared the responsibility of major user interface planning and design. The time gained here could be used in other aspects of the project. This, in practice, should significantly shorten the SDLC.
- The shortening of the SDLC could result in noticeable improvement in software engineering productivity, particularly for large projects. The hours gained in not having to program a user interface could be spent on other aspects of the project.
- The DMID not only addresses menu design, but user accessibility also. It is constructed in such a way as to enable the following constraints:
  - ✓ Only users who are defined to the DMID can gain access to the software system(s) employing it.
  - ✓ Through logical views, each user gets a picture of the system that is based the user's authorization log. Only those resources, to which the user is authorized, will show up on the user's display. So apart from not being able to access other resources, the user is given the illusion that they do not exist. Hence, user's perspective of the system is as narrow or broad as his/her span of authorization.
- The DMID is designed to support future changes in the functional and operational environment of the software system, without forcing a corresponding change in the underlying code. This will become clear as we proceed.

3. Basic Architecture of the Dynamic Menu Interface Designer

The concept of a DMID was first explored on a L*abor Market Information System (LMIS)* project (see [Foster, 1999]) with encouraging results. Since that time, a modern prototype of the project has been developed and tested, and is described here. We will discuss the DMID architecture from the perspectives of the database requirements and the user interface requirements.

3.1 Database Requirements

The basic architecture of the DMID calls for the use of a few relational tables which are described in figure 1. Notice that to aid subsequent referencing, each table is assigned a reference code (indicated in parentheses); attributes of each table are also assigned reference codes; all reference codes are indicated in square brackets; foreign key references are indicated in curly braces. Further, the relationship details of figure 1 are clarified in the entity-relationship diagram (ERD) of figure 2.

Figure 1: Normalized Relations for the DMID

System Definitions (E01): for storage of internal identifications of all information systems that use the DMID. Each system is assigned a unique identifier. Essential attributes include:
- System Code [SysCode]
- System Name [SysName]
- System Abbreviation [SysAbbr]
- Home Path [SysHome]

The primary key: {SysCode}

Participating Organizations (E02): for storage of internal identification(s) of the organization(s) that have access to the software system(s) of the host organization. Essential attributes include:
- Organization Code [OrgCode]
- Organization Name [OrgName]
- Organization Abbreviation [OrgAbbr]

The primary key: {OrgCode}

System Users (E03): for identification of all legitimate users of the system. Each user must belong to an organization that is recognized by the system. Essential attributes include:
- User Identification Code [UsrCode]
- User Login Name [UsrName]
- User First Name [UsrFName]
- User Last Name [UsrLName]
- User' Organization [UsrOrgCode] {**Refers to E02**}
- User Classification [UsrClass]
- User Password [UsrPssWrd]
- User Password Change Ceiling in days [UsrPssCeil]
- Date of Last Password Change [UsrPssChgD]

The primary key: {UsrCode}

System Operations (E04): for definition of all user operations (options) used. Essential attributes include:
- Operation Code [OpCode]
- Operation Implementation Name [OpIName]
- Operation Descriptive Name [OpDName]
- Operation Description [OpDscr]
- Operation Home Path [OprHome]

The primary key: {OpCode}

Figure 1: Normalized Relations for the DMID (continued)

System Menu Definitions (E05): for definition of all major menus and sub-menus used in the (possibly different) software system(s). Each menu is assigned a unique identifier, and is attached to an information system. Essential attributes include:

- Menu Code [MnuCode]
- Menu Implementation Name [MnuIName]
- Menu Descriptive Name [MnuDName]
- Menu Description [MnuDscr]
- Menu's System Code [MnuSysCode] {**Refers to E01**}
- Menu's Home Path [MnuHome]

The primary key: {MnuCode}

Menu Constituents (E06): the implementation of a M:M relationship between System Menu Definitions (E05) and System Operations (E04). Essential attributes include:

- Menu Code [MC_MnuCode] {**Refers to E05**}
- Menu Sequence Number [MC_MnuSeqN]
- Constituent Operation Code [MC_OpCode] {**Refers to E04**}

Candidate keys: {MC_MnuCode, MC_MnuSeqN}; {MC_MnuCode, MC_OpCode}

User Authorization to Operations (E07): the implementation of a M:M relationship between System Users (E03) and System Operations (E04). Essential attributes include:

- User Identification Code [AO_UsrCode] {**References E03}**
- Authorized Operation Code [AO_OpCode] {**References E04}**

The primary key: {AO_UsrCode, AO_OpCode}

User Authorization to Menus (E08): the implementation of a M:M relationship between System Users (E03) and System Menu Definitions (E05). Essential attributes include:

- User Identification Code [UM_UsrCode] {**Refers to E03**}
- Authorized Menu Code [UM_MnuCode] {**Refers to E05**}
- User Menu Sequence Number [UM_MnuSeqN]

Candidate keys: {UM_UsrCode, UM_MnuCode}; {UM_UsrCode, UM_MnuSeqN}

User Authorization to Systems (E09): the implementation of a M:M relationship between System Users (E03) and System Definitions (E01). Essential attributes include:

- User Identification Code [US_UsrCode] {**Refers to E03**}
- Authorized System Code [US_SysCode] {**Refers to E01**}
- User System Sequence Number [US_SysSeqN]

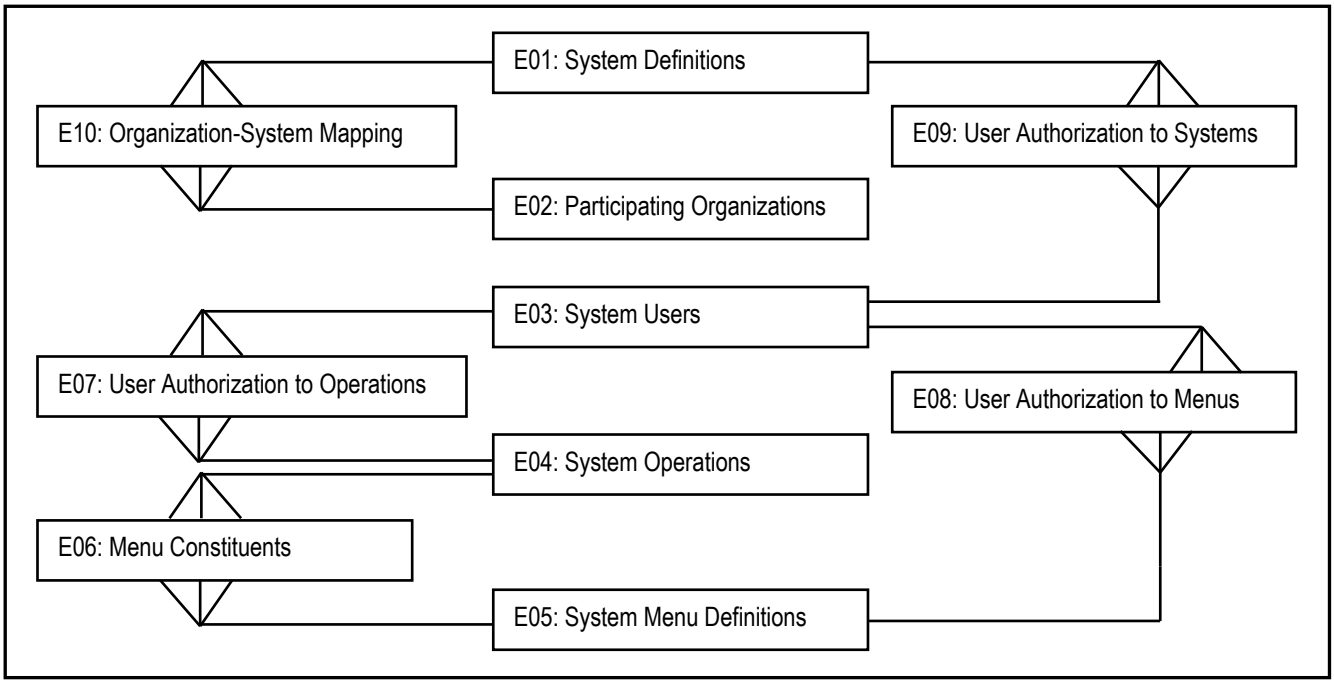
Candidate keys: {US_UsrCode, US_SysCode}; {US_UsrCode, US_SysSeqN}

Organization – System Mapping (O10): the implementation of a M:M relationship between Participating Organizations (E02) and System Definitions (E01). Essential attributes include:

- Organization Code [OS_OrgCode] {**Refers to E02**}
- System Code [OS_SysCode] {**Refers to E01**}
- System Sequence Number [OS_SysSeqN]

Candidate keys: {OS_OrgCode, OS_SysCode}; {OS_OrgCode, OS_SysSeqN}

Figure 2: Entity-relationship Diagram for the DMID



Through these relational tables, one can accurately and comprehensively define the constituents, structure, and security constraints of the user interface for any software system that requires the use of user menu(s). Specifically, here is a summary of information that could be specified for each software system:

- Definitional details for the software system(s) employing the DMID
- Basic information about the participating organization(s)
- Basic information about users who will access their respective software system(s) via the DMID
- Definitional details for the operations that comprise each participating software system
- Definitional details for the menus and submenus for each participating software system
- The structure and operational constituents of each menu/submenu comprising each participating software system
- User authorization matrix for subsystems comprising each participating software system
- User authorization matrix for operations comprising each participating software system
- User authorization matrix for participating software systems
- Mapping of participating system(s) for each participating organization (particularly relevant if there are multiple participating organizations and multiple systems)

To facilitate the users having different perspectives of the software system(s), based on their authorization schedule, a number of logical views are required; the salient ones are described in figure 3.

Figure 3: Important Logical Views for the DMID

User's System Overview [DM_UsrSysA_LV1]: This is the logical join of User-System Authorizations (E09) with System Definitions (E01), and System Users (E03). Attributes will be read-only:

- User Identification Code [US_UsrCode]
- User Login Name [UsrName]
- User First Name [UsrFName]
- User Last Name [UsrLName]
- System Code [US-SysCode]
- User System Sequence Number [US_SysSeqN]
- System Name [SysName]
- System Abbreviation [SysAbbr]

User Menus Summary [DM_UsrMnuA_LV1]: This is the logical join of User-Menu Authorizations (E08) with System Menu Definitions (E05) and System Users (E03), and System Menu Definitions (E05) with System Definitions (E01). Attributes will be read-only:

- User Identification Code (UM_UsrCode)
- User Login Name [UsrName]
- User First Name [UsrFName]
- User Last Name [UsrLName]
- Menu Code (UM_MnuCode)
- Menu Implementation Name [MnuIName]
- Menu Descriptive Name [MnuDName]
- Menu's System Code [MnuSysCode]
- System Name [SysName]
- System Abbreviation [SysAbbr]
- User Menu Sequence Number (UM_MnuSeqN)

User Operations Summary [DM_UsrOprA_LV1]: This is the logical join of User-Operation Authorization (E07) with Menu Constituents (E06), System Operations (E04), System Users (E03), and Menu Definitions (E05). Attributes will be read-only:

- User Identification Code [UO_UsrCode]
- User Login Name [UsrName]
- User First Name [UsrFName]
- User Last Name [UsrLName]
- Authorized Operation Code [UO_OpCode]
- Operation Implementation Name [OpIName]
- Operation Descriptive Name [OpDName]
- Menu Code [MC_MnuCode]
- Menu Sequence Number [MC_MnuSeqN]
- Constituent Operation Code [MC_OpCode]
- Menu Implementation Name [MnuIName]
- Menu Descriptive Name [MnuDName]

Organization-System Mapping [DM_OrgSysM_LV1]: This is the logical join of Organization-System Mapping (E10) with Participating Organizations (E02), and System Definitions (E01). Attributes will be read-only:

- Organization Code [OS_OrgCode]
- Organization Name [OrgName]
- Organization Abbreviation [OrgAbbr]
- System Sequence Number [OS_SysSeqN]
- System Code [OS_SysCode]
- System Name [SysName]
- System Abbreviation [SysAbbr]

Figure 3: Important Logical Views for the DMID (continued)

System Users [DM_User_LV1]: This is the logical join of System Users (E03) with Participating Organizations (E02). Attributes will be read-only:

- User Identification Code [UsrCode]
- User Login Name [UsrName]
- User First Name [UsrFName]
- User Last Name [UsrLName]
- User' Organization [UsrOrgCode]
- Organization Name [OrgName]
- Organization Abbreviation [OrgAbbr]
- User Classification [UsrClass]
- User Password [UsrPssWrd]
- User Password Change Ceiling in days [UsrPssCeil]
- Date of Last Password Change [UsrPssChgD]

System Menu Definitions [DM_MenuD_LV1]: This is the logical join of System Menu Definitions (E05) and System Definitions (E01). Attributes will be read-only:

- Menu Code [MnuCode]
- Menu Implementation Name [MnuIName]
- Menu Descriptive Name [MnuDName]
- Menu Description [MnuDscr]
- Menu's System Code [MnuSysCode]
- System Name [SysName]
- System Abbreviation [SysAbbr]

System Menu Constituents [DM_MenuC_LV1]: This is the logical join of System Menu Constituents (E06), Menu Definitions (E05), System Definitions (E01), and System Operations (E04). Attributes will be read-only:

- Menu Code [MC_MnuCode]
- Menu Implementation Name [MnuIName]
- Menu Descriptive Name [MnuDName
- Menu's System Code [MnuSysCode]
- System Name [SysName]
- System Abbreviation [SysAbbr]
- Menu Sequence Number [MC_MnuSeqN]
- Constituent Operation Code [MC_OpCode]
- Operation Implementation Name [OpIName]
- Operation Descriptive Name [OpDName]

Key:

- Attributes in black are taken from the reverencing relation
- Attributes in blue are taken from the referenced relation(s)

With these logical views, the next step is to superimpose on the database, a user interface that provides the user with the advantages mentioned earlier.

3.2 User Interface Requirements

Having described the database requirements, let us now examine the user interface requirements for the DMID. This user interface will be superimposed on top of the relational database. Through the DMID user interface, it must be possible to define and configure the user interface for any software system. Following are some basic guidelines.

From the outset, the DMID project was designed to meet the following minimum objectives:

- **Usefulness:** Software engineers must be able to use the DMID as a means of shortening the development time on software systems that they have under construction.
- **Interoperability:** The DMID must be applicable across various software systems and operating systems platforms.
- **User-friendliness:** The system must be user friendly and easy to use. The interface must be intuitive so that there is a very short learning curve.
- **Reusability:** It must be possible to use the DMID on various independent software systems, as well as a conglomeration of software systems operating as part of a larger integrated system.
- **Flexibility:** The DMID must provide users with the flexibility of specifying the relative order of menu options comprising the system being constructed.

3.2.1 Logging On

The user's first interaction with the DMID is via logging on. In logging on, the user specifies the following:

- User Identification Code or Name
- Organization Code or Name
- Password (not displayed in the interest of security)

This information is checked against the internal representations stored in the database. If a match is found, the user is taken to the next stage; otherwise, the user is given an appropriate error message, and allowed to try logging on again (the number of allowable attempts may be appropriately restricted).

In its initial configuration, two classes of users are facilitated — end users and system administrators. System administrators have access to the Administrative Specification Management (ASM) subsystem. This subsystem provides select, update, deletion, and insertion (SUDI) privileges to all the data contained in the internal database. This means the administrator can carry out functions of defining the operational requirements and constraints of software systems, subsystems, and users (as described in the upcoming subsection). End users have access to the End-user Access Control (EAC) subsystem (elaborated shortly).

3.2.2 Management of System Constraints

The ASM subsystem provides facilities for defining, reviewing, and modifying the operational scope of participating systems using the DMID, as well as the operational constraints for system users. Among the related activities for this feature are the following:

- Addition of new system(s) and subsystems
- Deletion of (obsolete) system component(s)
- Addition of new operations and/or menu options
- Deletion of menu options
- Configuring/restructuring of system menus
- Setting user authorization schedules in respect of access to systems, subsystems, and operations
- Change of user authorization schedules in respect of access to systems, subsystems, and operations
- Addition of new users
- Deletion of users
- Setting the Organization–System Mapping
- Changing the Organization–System Mapping
- Reviewing (querying) system constraints information

Figure 4 shows a screen capture from the ASM subsystem. In this illustration, a user called Lambert is working with system definitions. At first entry into this option, an initial list of all software systems being managed through the DMID is provided. As these system definitions are modified, or if items are removed during the session, the list is updated. New entries can also be made. Finally, notice that there is a **Search** button to the bottom right of the panel. If this is clicked, a related operation is invoked as illustrated in figure 5. This will allow the user (in this case Lambert) to peruse through the system definitions using any combination of the search criteria provided.

It should be noted that the illustrations of figures 4 and 5 represent the most basic operational features of the ASM subsystem. There are other more sophisticated operations involving access of the logical views of section 3.1 (revisit figure 2). Figure 6 provides an example. Here, user-menu authorizations are being managed; specifically, user Lambert is perusing a logical view that joins multiple tables (review the spec for **User Menus Summary [DM_UsrMnuA_LV1]** in figure 3**)**. He has the option of listing all user-menu combinations stored, or narrowing the search by specifying data for any of the six search criteria shown. However, please note that each operation follows the same design concept.

Only individuals who are duly authorized to carry out the functions of system configuration and management will have access to the ASM subsystem; they must have the *administrator* classification.

Figure 4: Screen-shot from the DMID's ASM Subsystem — Managing System Definitions

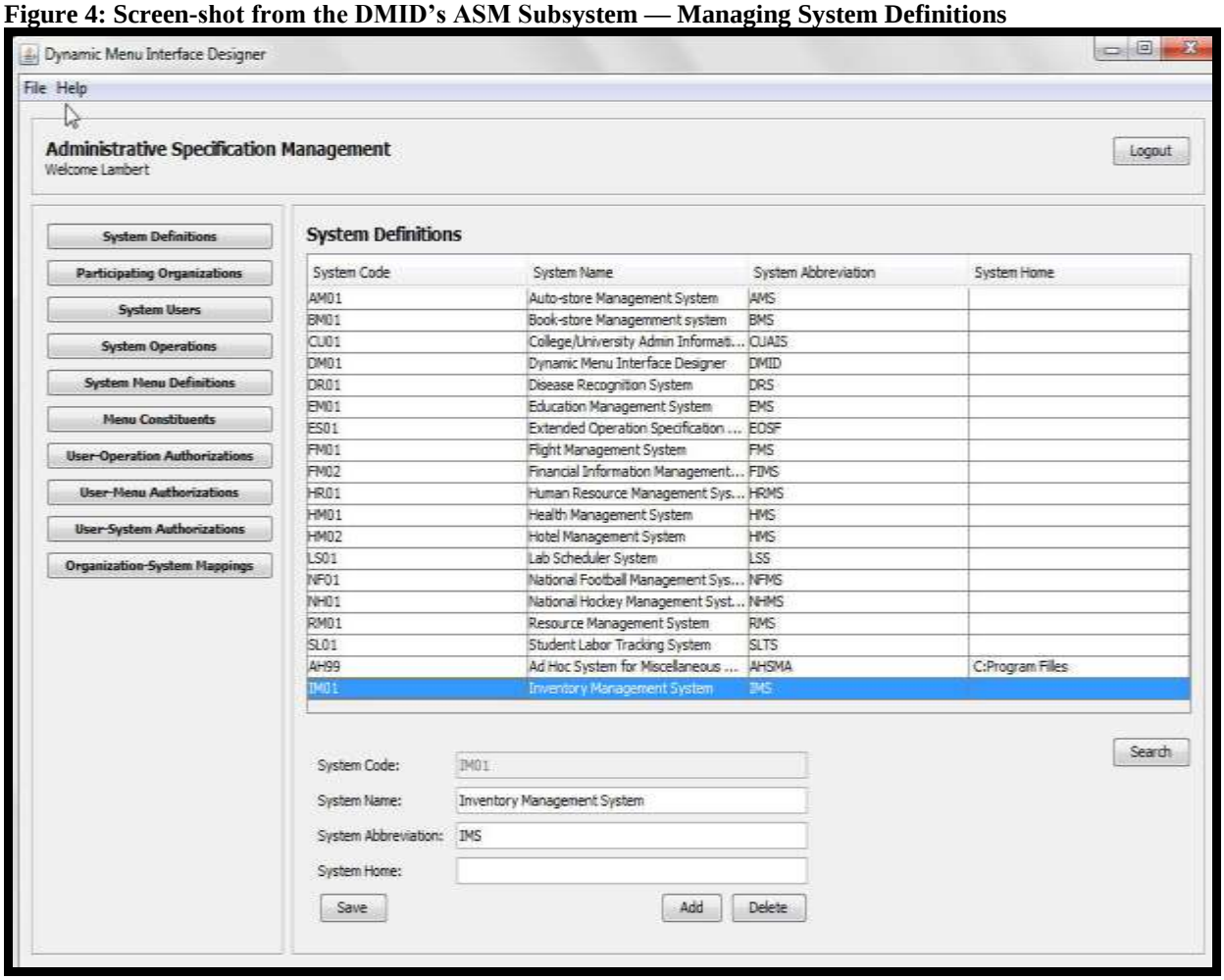


Figure 5: Screen-shot from the DMID's ASM Subsystem — Searching on System Definitions

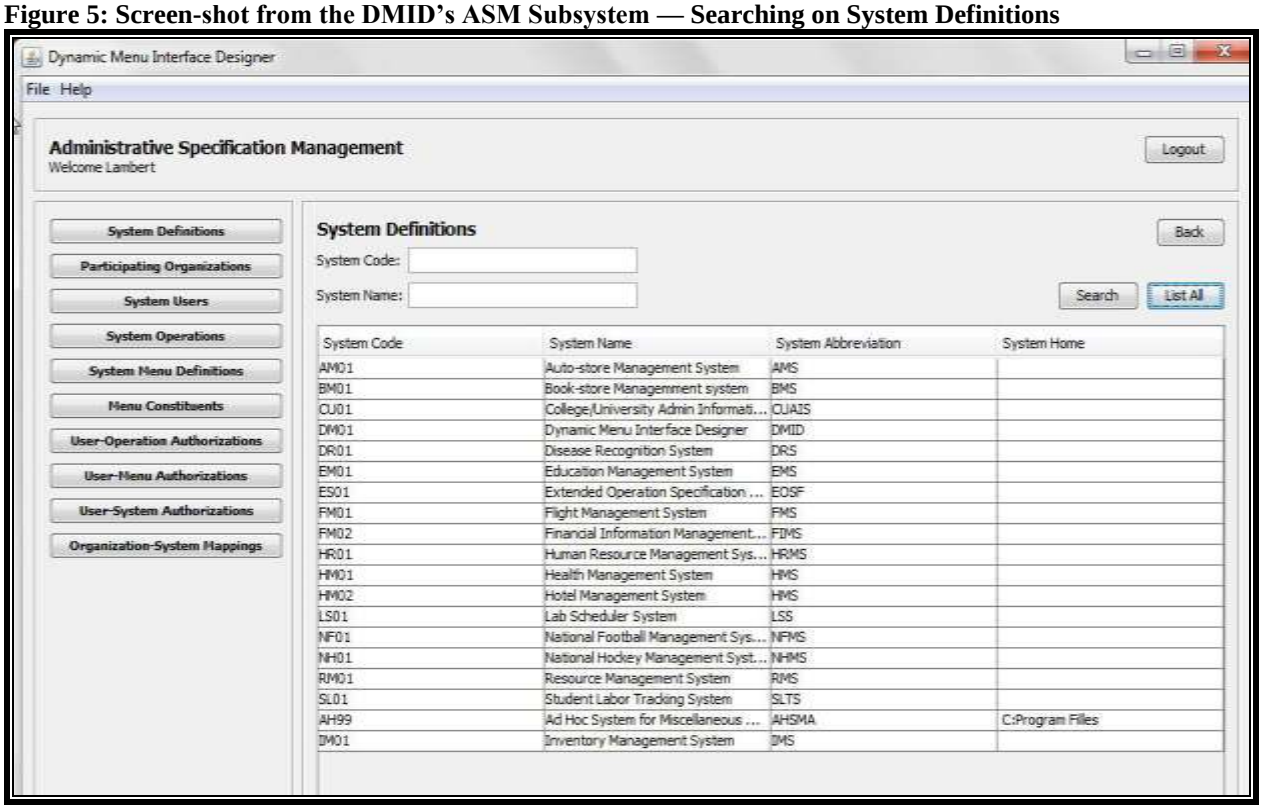
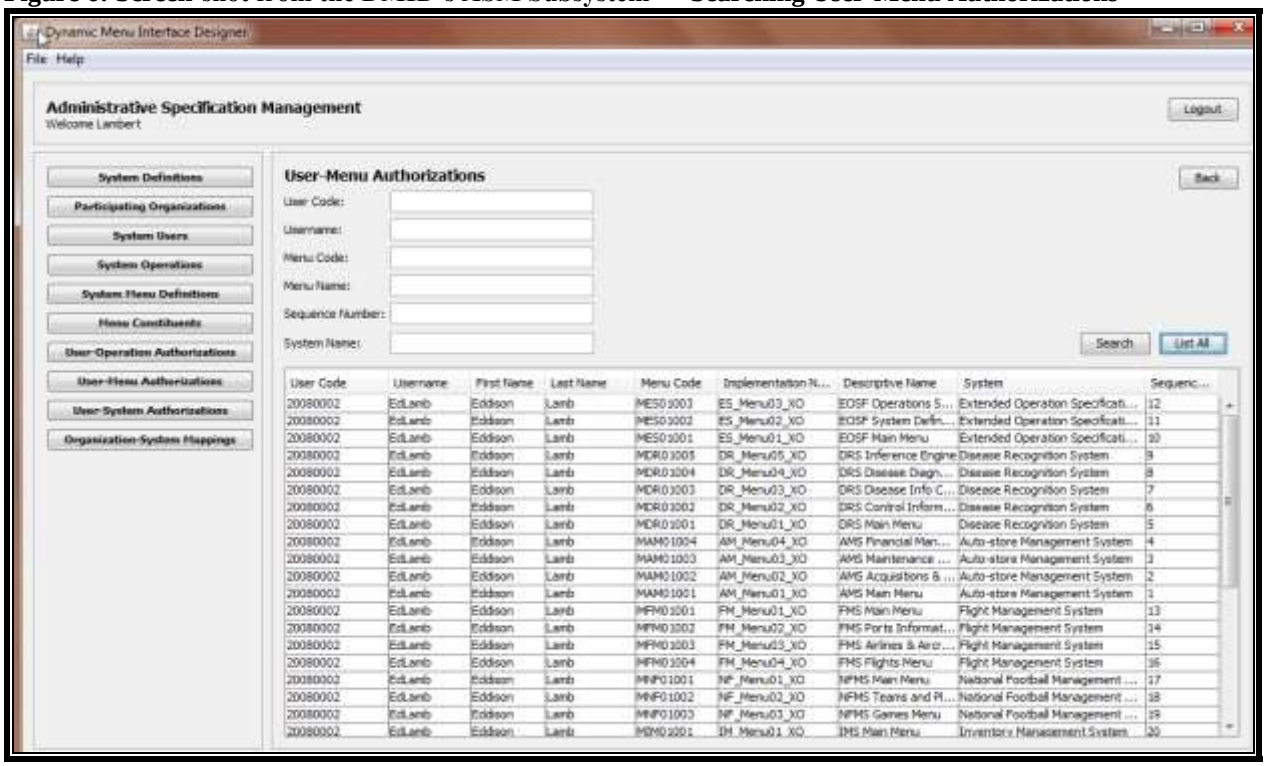


Figure 6: Screen-shot from the DMID's ASM Subsystem — Searching User-Menu Authorizations



3.2.3 Access to System Resources

Let us now revisit the EAC subsystem. Through this subsystem, an end user can only perform functions defined by his/her authorization schedule; in fact, only these capabilities will appear on his/her menu.

Assuming successful logon, the user gets a menu, depending on his/her authorization schedule that is stored in the underlying DMID database. Three mutually exclusive scenarios are likely:

a. Being an end user, the user gets a menu with the software system(s) to which he/she has access rights.
b. The end user is provided with a blank menu, representing zero access to resources.
c. The user, being a system administrator, gains access to the ASM menu.

From here on, the user's display panel varies according to what system resources he/she is authorized to access; only resources to which the user is authorized are shown on his/her display panel. Figures 7 and 8 illustrate two extremes: In figure 7a, the main menu for a user called Edison is shown; notice that Edison has access to all the software systems managed via the DMID, but has selected the "Auto-store Management System." In figure 7b, the user (in this case Edison) is provided with a submenu of subsystems of the previously selected system, to which he/she has access privileges. Then, in figure 7c, after selecting the "Acquisitions and Sales" subsystem from the previous screen, another submenu of related operations to which the user is authorized is provided. At this point, the user may select any desired operation for execution. At each level, the user's display panel is filled with options so that he/she is able to scroll and select the option of choice (by highlighting and clicking the **Select** push button).

Figure 7a: Screen-shot from the DMID's EAC Subsystem — Main Menu

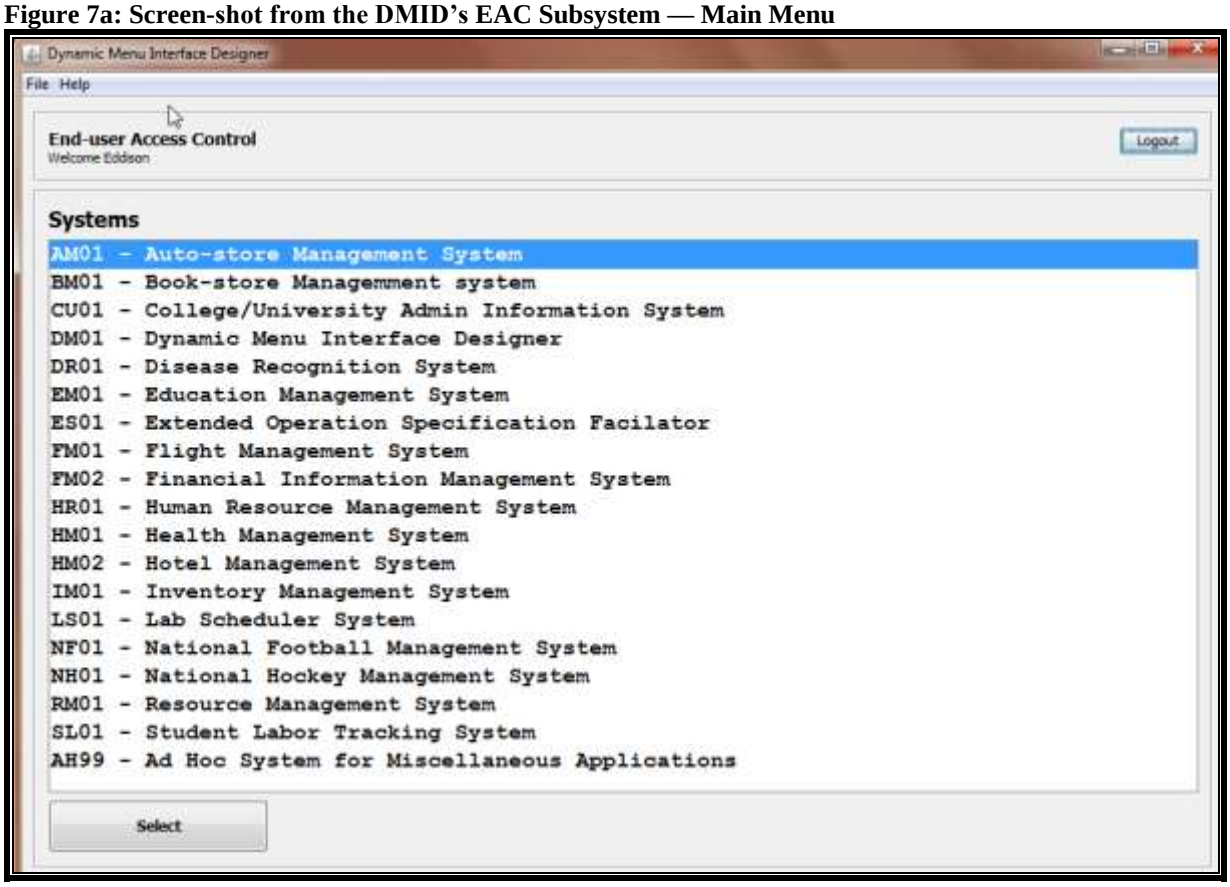


Figure 7b: Screen-shot from the DMID's EAC Subsystem — Subsystems Menu

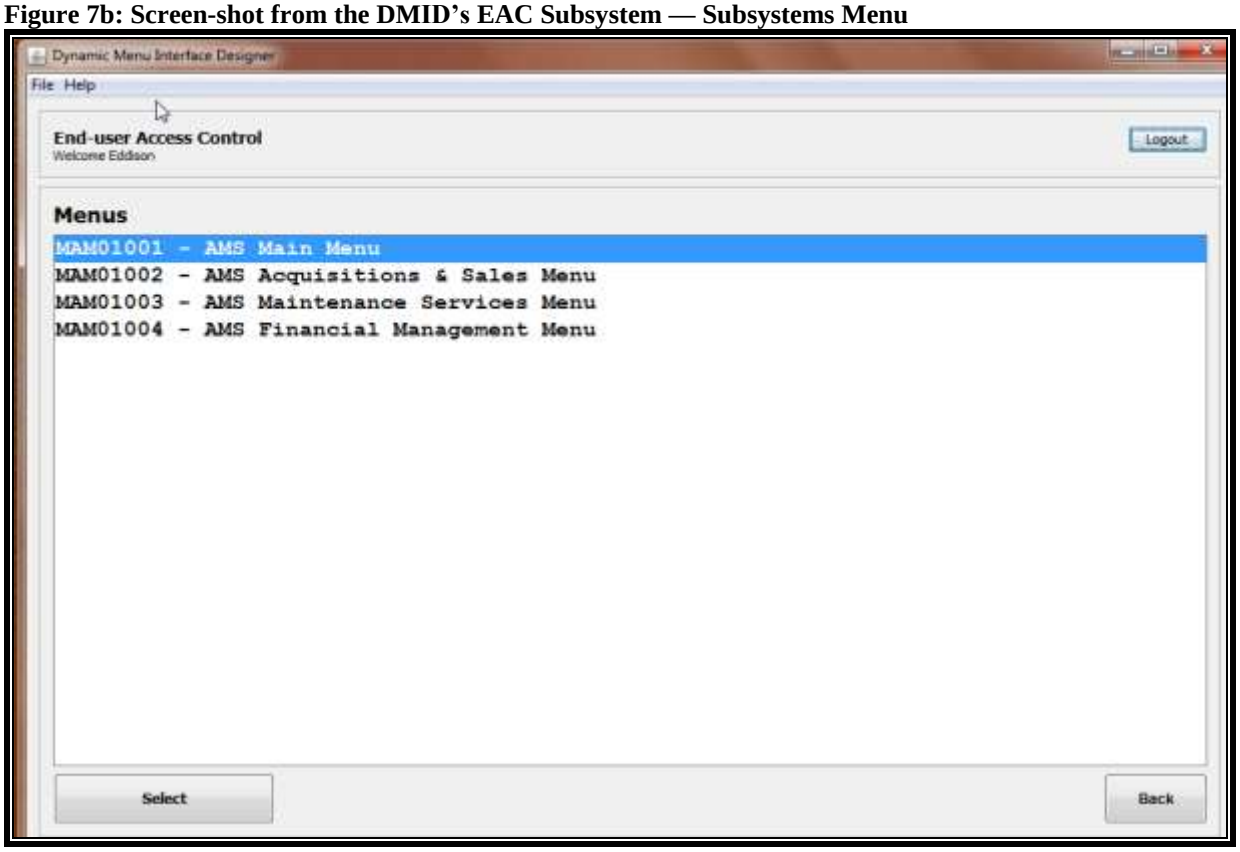
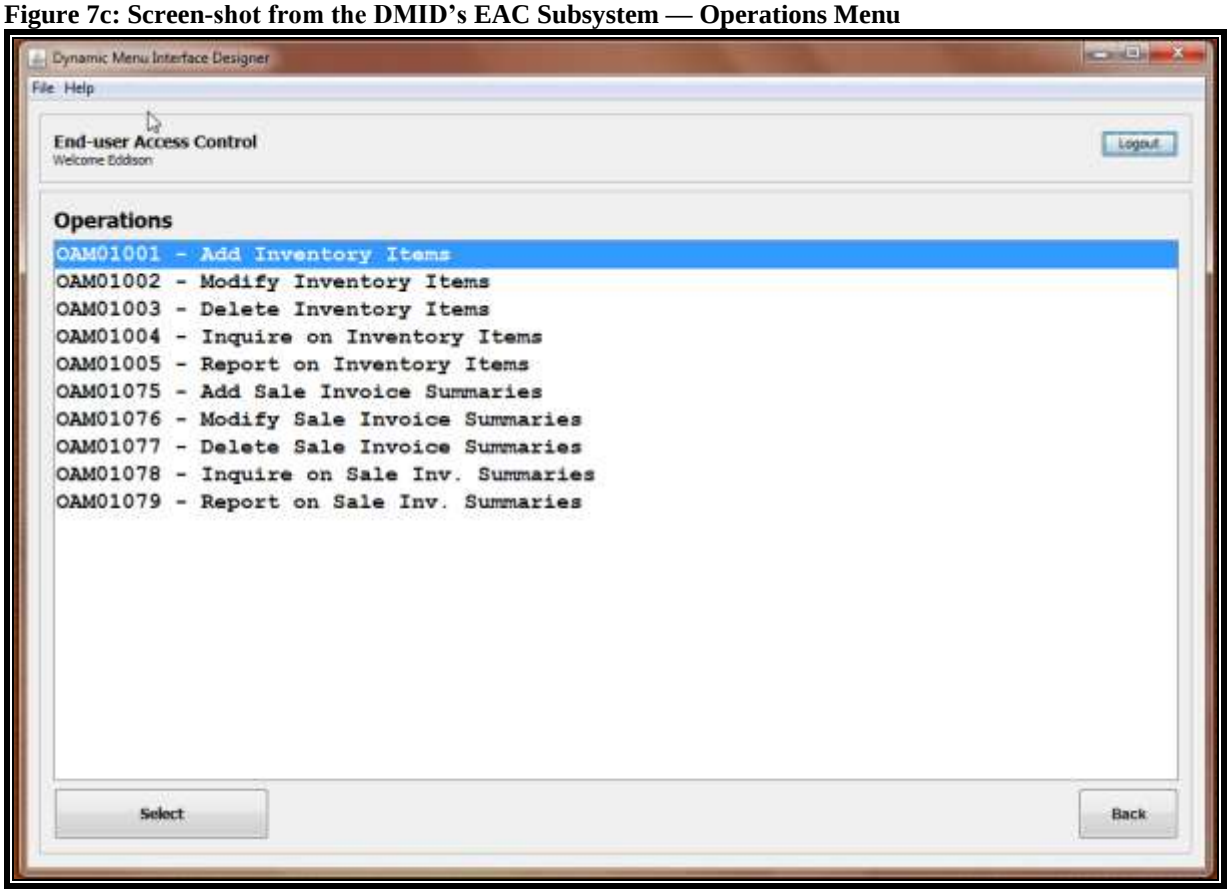


Figure 7c: Screen-shot from the DMID's EAC Subsystem — Operations Menu



In figures 8a – 8c, the user (in this case Ann Marie) has more limited access to systems, subsystems, and operations managed by the DMID. In the most extreme scenario, a user may have no access to any resource (system, subsystem, or operation), in which case a blank screen would appear, and the user would not be able to do anything.

Figure 8a: Screen-shot from the DMID's EAC Subsystem — Main Menu

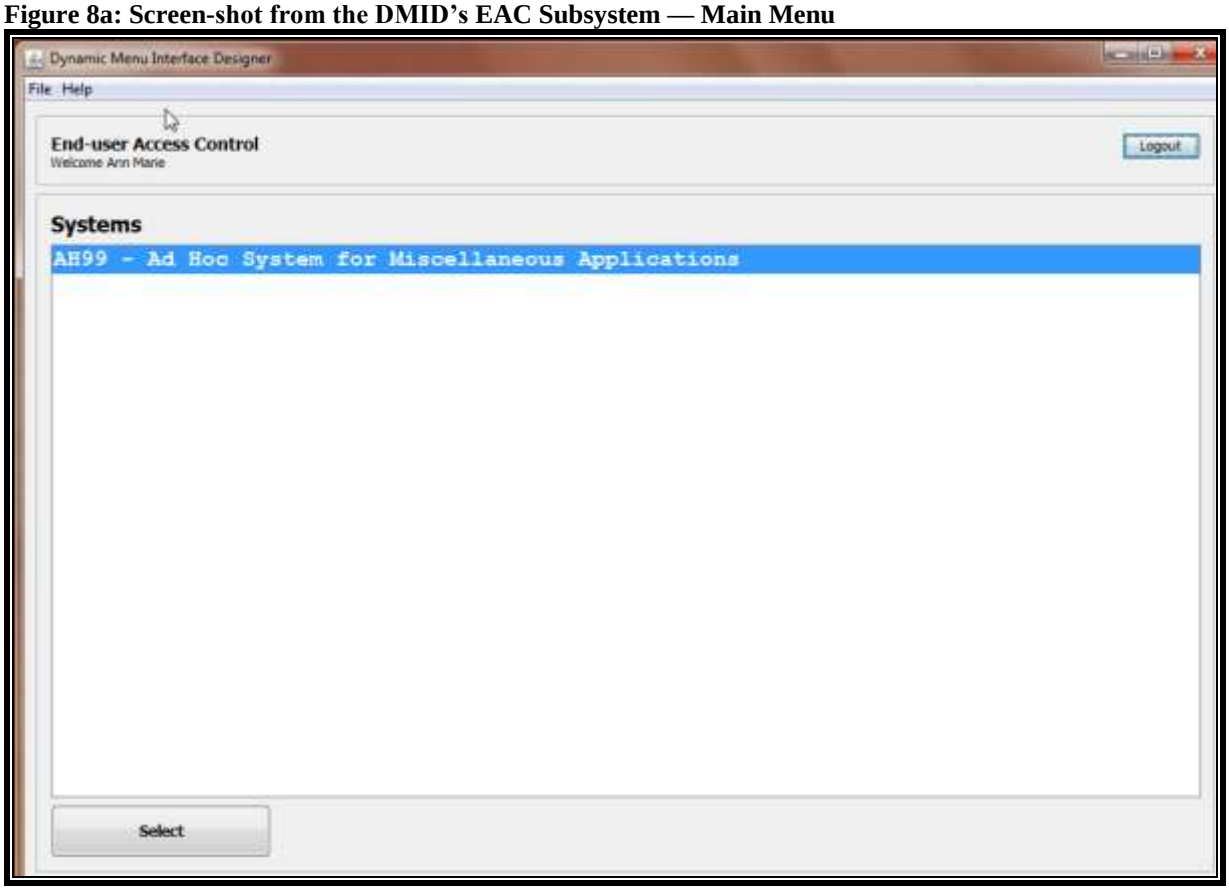


Figure 8b: Screen-shot from the DMID's EAC Subsystem — Subsystems Menu

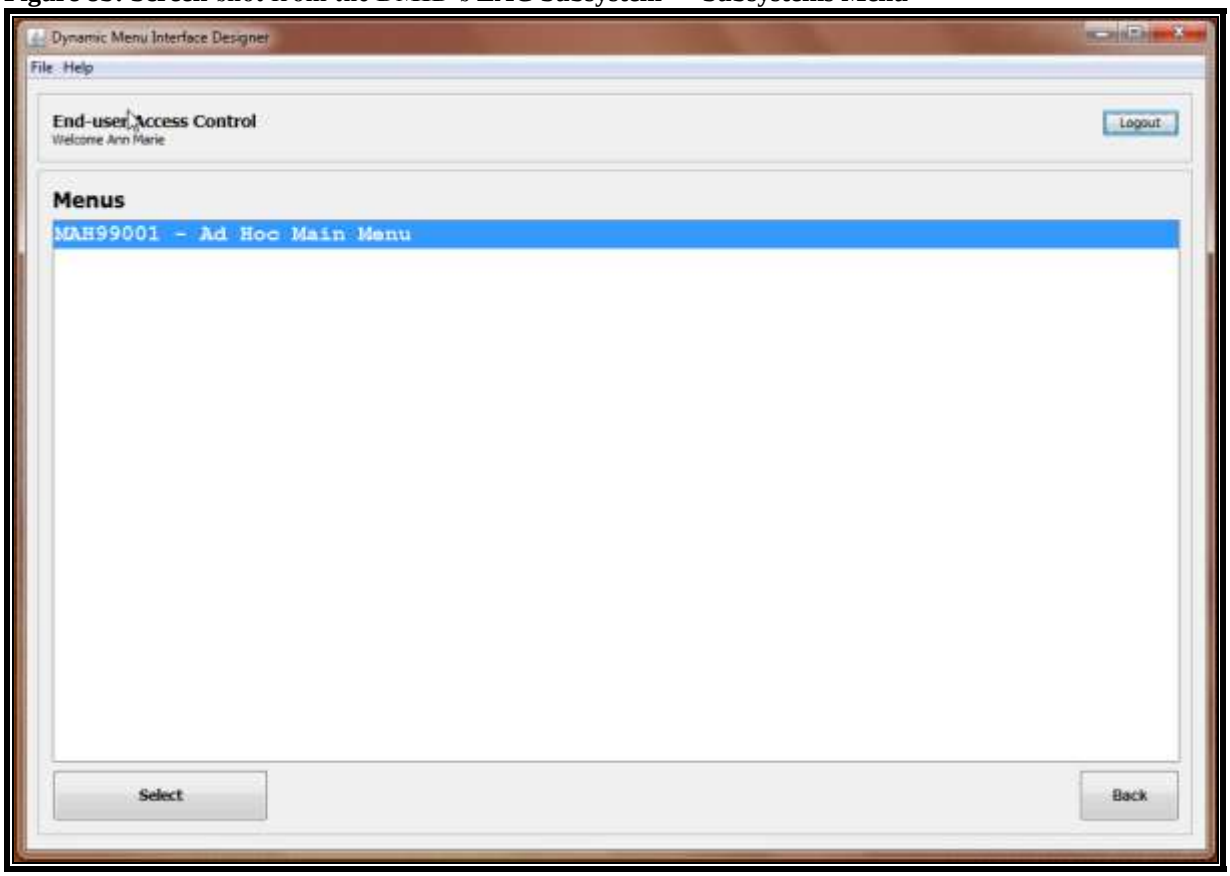
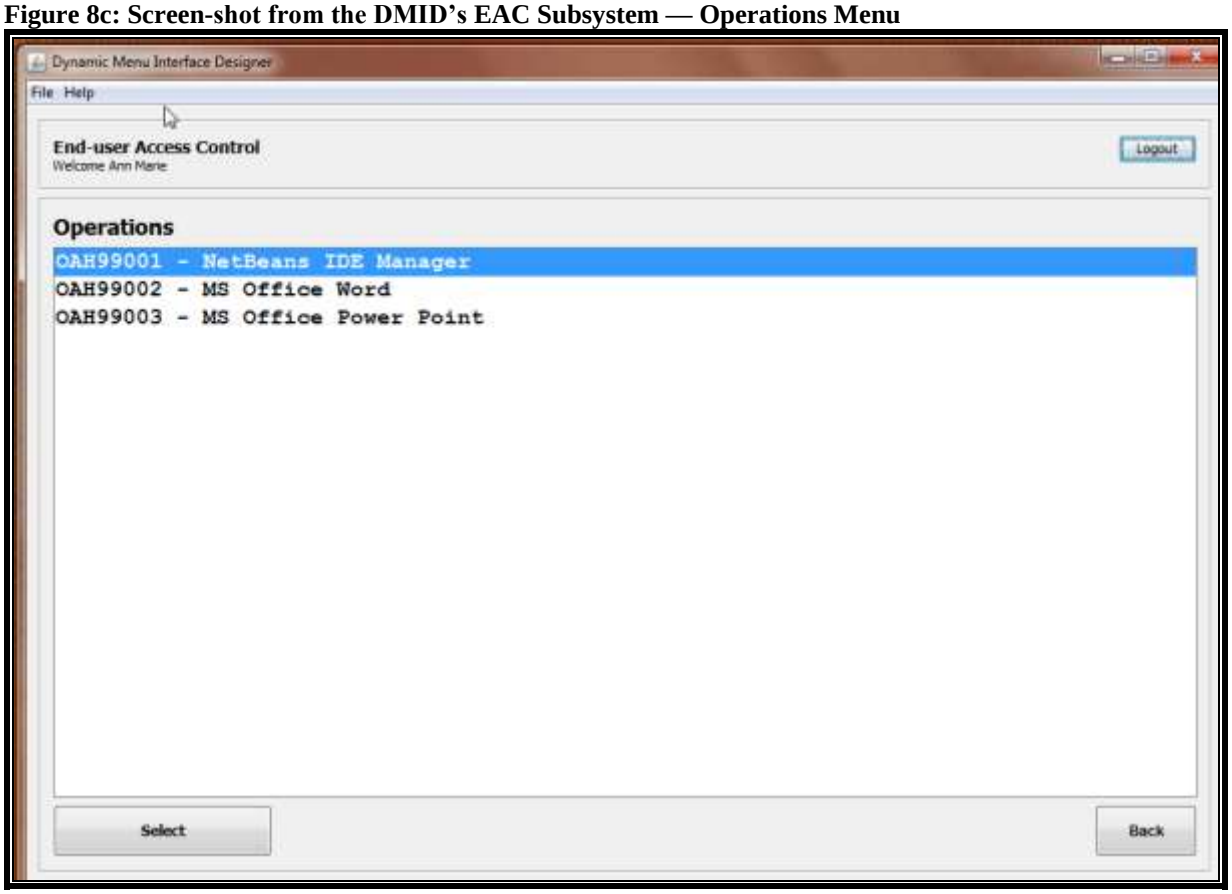


Figure 8c: Screen-shot from the DMID's EAC Subsystem — Operations Menu



4. DMID Prototype

The previous section provided information drawn from a DMID prototype that was developed and tested for the purpose of this project. Influenced by user requirements such as interoperability and reusability (review section 3.2), as well as affordability, the prototype was developed using a MySQL database on the backend, and Java NetBeans as the frontend integrated development environment (IDE). These development tools are readily available (free of charge) for the major operating system platforms.

The prototype consists of 29 integrated operations (application programs) spread over approximately 12,280 lines of code. It is also accompanied by a comprehensive design specification.

For obvious reasons related to user-friendliness, the DMID user interface is graphical (but excluding the drag-and-drop feature). Learning it is therefore straightforward and intuitive. The prototype has been tested with sample data that was specifically developed for that purpose. We are currently in the process of testing it with representative sample data for various working environments.

5. Summary and Concluding Remarks

This paper has made the case for a dynamic menu interface designer (DMID) as a software component that can be used on multiple software engineering projects. It has presented the basic architecture of such as a software system, and has described a successful prototype of the system.

Given the significance of the promise for improved productivity, usefulness, interoperability, user-friendliness, reusability, and flexibility, the DMID project certainly deserves additional research and attention. We are giving the initiative this attention, and are encouraged by the prospects. When one considers that we are in an era where there is great need for ubiquitous, reusable software components (see [Dey, 1997], [Martin, 1993], and [Niemelä and Latvakoski, 2004]), having such a product would certainly bring some comfort to software engineers, information systems managers, and end users.

References

[AntsSoft, 2009] AntsSoft. *ANTSSOFTUltra Menu*. AntsSoft, 2000 – 2009. http://www.antssoft.com/ultramenu/index.htm (accessed July 2010).

[Chan, 1998]: Chan, H., et. al., "The Effect of Data Model, System and Task Characteristics on Query Performance – An Empirical Study", *The Database, Winter, 1998, Volume 29, Number 1,* 31-46, 1998.

[Curl, 1998] Curl, S., et. al., "An Investigation of the Roles of Individual Differences and Interface on Data Usability ", *The Database, Winter, 1998, Volume 29, Number 1,* 50-64, 1998.

[Dey, 1997] Dey, A. K., Abowd, G. and Pinkerton, M. "CyberDesk: A Framework for Providing Self-Integrating Ubiquitous Software Services" *GVU Technical Report, GIT-GVU-97-10*. http://www.cc.gatech.edu/fce/cyberdesk/pubs/UIST97/UIST97.html (accessed May 2011).

[Drupal, 2010] Drupal. *Drupal Documentation*. http://drupal.org/handbook (accessed July 2010).

[Foster, 1999] Foster, Elvis C. *Labour Market Information System: Thesis*, Kingston, Jamaica: Department of Mathematics and Computer Science, University of the West Indies, 1999.

[Foster, 2010] Foster, Elvis C. *Software Engineering: A Methodical Approach.* Bloomington, IN: Xlibris Corporation, 2010. See chapter 11.

[Glass, 1998] Glass, R. "Short-term and Long-term Remedies for Runaway Projects", *Communications of the ACM, July, 1998, Volume 41, Number 7,* 13-15, 1998.

[Khalifa, 1998] Khalifa, M. "Computer Assisted Evaluation of Interface Design", *The Database, Winter, 1998, Volume 29, Number 1,* 66-81, 1998.

[Kivisto, 2000] Kivisto, Kari. *A Third Generation Object-Oriented Process Model*. Department of Information Processing Science, University of Oulu, 2000. http://herkules.oulu.fi/isbn9514258371/isbn9514258371.pdf (accessed July 2010). See chapter 3.

[Martin, 1993] Martin, J. and Odell, J. *Principles of Object Oriented Analysis and Design.* Eaglewood Cliffs, New Jersey: Prentice Hall, 1993. See chapters 1 – 3.

[Microsoft, 2010a] Microsoft. *Windows Server*. Microsoft, 2011. http://technet.microsoft.com/en-us/windowsserver/default.aspx (accessed July 2010).

[Microsoft, 2010b] Microsoft. *How To: Use Authorization Manager (AzMan) with ASP.NET 2.0*. Microsoft, 2011. http://msdn.microsoft.com/en-us/library/ff649313.aspx (accessed July 2010).

[Microsoft, 2010c] Microsoft. *Windows Presentation Foundation*. Microsoft, 2011. http://msdn.microsoft.com/en-us/library/ms754130.aspx (accessed July 2010).

[Nielsen, 1993] Nielsen, J., *Usability Engineering.* Boston: Academic Press, 1993.
[Nielsen, 1999] Nielsen, J., "User Interface Directions for the Web", *Communications of the ACM, January, 1999, Volume 42, Number 1*, 65-72, 1999.

[Niemelä and Latvakoski, 2004] Niemelä, E. and Latvakoski, J. "Survey of Requirements and Solutions for Ubiquitous Software", *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia MUM 04 (2004)*, ACM Press, 71-78.

[Robillard, 1999] Robillard, P. "The Role of Knowledge in Software Development", *Communications of the ACM, January, 1999, Volume 42, Number 1*, 87-92, 1999.

[Schneiderman, 2005]: Schneideman, Ben., *Designing the User Interface 4th ed*. Reading, MA: Addison-Wesley, 2005.