Distributing a Control Room Using PHP and HTML

Jørn G. Larsen, Kenneth R. Sørensen, Mikael Bo Andersen
Martin Nielsen, Gunnar B. Sigurðarson, Svend Møller

01gr733

This paper deals with the development of a control room where PHP: Hypertext Pre-processor (PHP) scripts, HyperText Markup Language (HTML) and the World Wide Web (WWW) are used to distribute data worldwide. The focus is on security problems that arise when an Internet browser is used as client software. This includes: Validate that the control room is used correctly, how to protect sensitive data, identification and verification of the users. The problems encountered are general even though a specific analysis and design of a distributed control room for a small-satellite forms the basis for the research. It is concluded that the technologies chosen, are capable of offering a security level, which is acceptable in many scientific and industrial applications.

Distributed Control Room

In many industries, there is a need for a control room from where an employee is capable of supervising systems and processes. Until recently, this has often been implemented in one central physical location, requiring a large facility, which is quite expensive.

As network technologies have improved and the Internet has become common, the foundation for distribution of data between separate locations has improved. Traditionally an implementation of a distributed control room includes both dedicated server and client software.

With the development and wide spreading of WWW, distribution is even easier. Everybody with access to the WWW has a web browser, thus there is no need for dedicated client software. An example of such a system is the JSWITCH developed by NASA [JSWITCH]. This system uses intelligent Java™ applets to access a distributed control room.

NASA had a few problems regarding the platform independence of Java™ applets. In this project, the client is made simpler to reduce these problems. HTML is used at the client side and PHP is used at the server side. This means that all intelligence is moved to the server.

Control Room for a Small Satellite

When developing a distributed control room it is essential to know how the users interact with the system. Figure 1 depicts users and their access to tasks in a small-satellite control room.

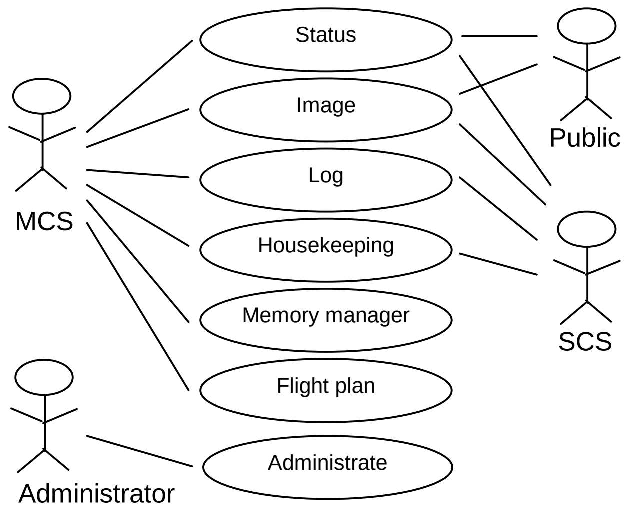

Figure 1 Use case diagram

The control room has four groups of users: The Mission Control Staff (MCS) is controlling the satellite and has access to all parts of the system regarding the satellite. The Scientific Staff (SCS) is allowed to view information about the satellite and to control the payload of the satellite. Public users are allowed to view some information about the satellite. Finally, administrators are able to manage the users of the system.

Furthermore, the figure illustrates that the users have access to seven different tasks, where the first one is status. This task nicely presents key information about the satellite. The task image is used to view images taken by the onboard camera of the satellite. With the log and housekeeping tasks, users are able to get a detailed view of the behaviour of the satellite. Software updates and memory dumps are carried out in the task memory manager. The flight plan is used when scheduling tasks, which has to be performed by the satellite. The last task is the administrate task which is used to control the users' access to the system.

When HTML and PHP is used to implement the system described above some central problems, arise.

- **User interaction**: When not using dedicated client software, the user is able to jump from one web page to another and therefore it is hard to control the sequence of the user interaction.
- **Access control**: User access to parts of the system must be restricted.
- **Secure communication**: Sensitive information in transit between browser and server has to be protected.

User Interaction

A way to improve the knowledge about the users' interaction with the system is to describe the tasks by use cases. The use cases specify states of the system and functions causing transitions between these.

Figure 2 is an example of a use case diagram. It shows how an administrator manages users of the control room.

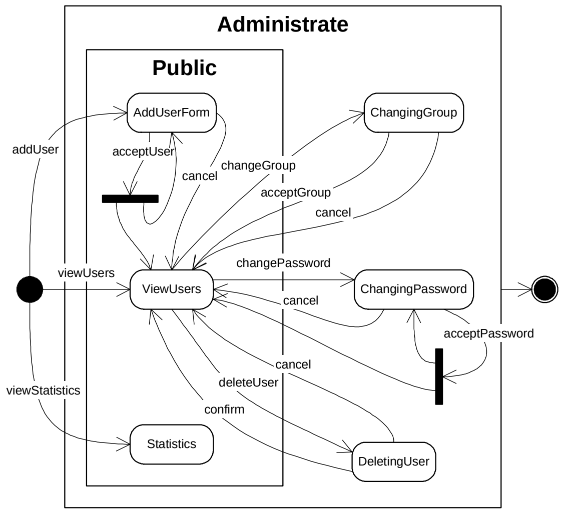

Figure 2 State diagram for Administrate use case

The main problem in web applications is that users are able to access web pages in any order. Consequently, it is difficult to control their behaviour. In traditional software, the developer is able to control the interaction between the user and the application by the design of the Graphical User Interface (GUI).

To solve this problem, a use case validation system is implemented, which guarantees that the user navigates correctly through the system. This includes two aspects: describing the legal use and monitoring the actual use.

Representing Use Cases

The use cases are stored in a table in a database. Each entry in the table represents a valid state transition. A valid transition is described by four fields. The first field specifies the class containing the function causing the transition. The second field specifies the function. The third field specifies the required system state before execution and the last field specifies the state obtained after execution. The current state field can represent a specific state or the "any" state. If the required state is "any", the function may be executed from any system state.

Converting Figure 2 to a database representation results in the following entries described in Table 1.

Class	Function	Current state	New state
Admin	AddUser	Any	AddUserForm
Admin	ViewUSers	Any	ViewUsers
Admin	ViewStatistics	Any	Statistics
Admin	AcceptUser	AddUserForm	ViewUsers
Admin	ChangeGroup	ViewUsers	ChangingGroup
Admin	AcceptGroup	ChangingGroup	ViewUsers
Admin	ChangePassword	ViewUsers	ChangingPassword
Admin	AccpetPassword	ChangingPassword	ViewUsers
Admin	deleteUser	ViewUsers	DeletingUser
Admin	Confirm	DeletingUser	ViewUsers
Admin	Cancel	AddUserForm	ViewUsers
Admin	Cancel	ChangingGroup	ViewUsers
Admin	Cancel	ChangingPassword	ViewUsers
Admin	Cancel	DeletingUser	ViewUsers

Table 1 Administrate use case represented in database

The first three entries are the public part of the use case and are executable from any present state. The six following entries specify six functions, which may only be called from one state each. The remaining four entries indicate that the function cancel may be executed from four different states.

Use Case Validation

To monitor and validate the behaviour of the users, the current state of each user session is stored using the session handling abilities in PHP. Code example 1 shows how a session is started and a session variable is registered.

```
session_start();
session_register('systemstate');
```

Code example 1

By doing this, the variable systemstate is available in any script executed by the user from until the session is ended.

Whenever a function is executed, the validation system checks the database to find out if the system is in a state where this is allowed. If it is, the function executes and the state of the system is updated. Otherwise, the user is informed about the illegal use. Figure 3 shows the use case validation procedure.

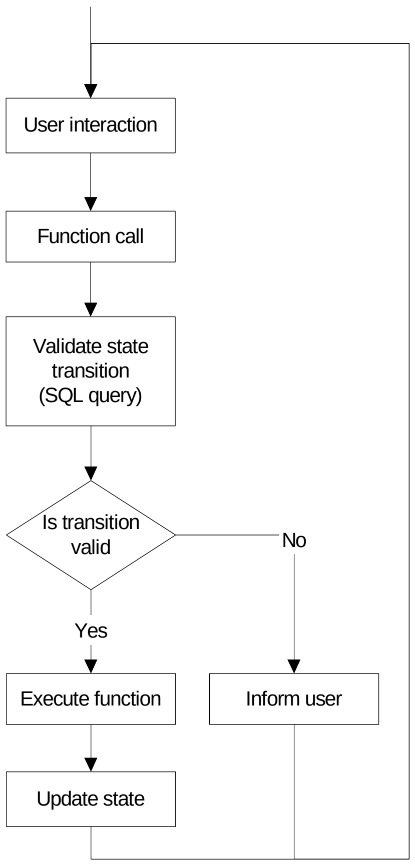

Figure 3 Flowchart of use case validation

Access Control

To manage the users' access to the different tasks an access control system is implemented. Four aspects are considered: How to represent different groups in the system, how to limit the number of active users in a task, how to handle concurrent logins by a user, and how to handle improper logouts.

The users of the system are divided into four groups. Each group has a set of tasks associated. The groups are organized in a hierarchical structure. The structure is depicted in figure 4.

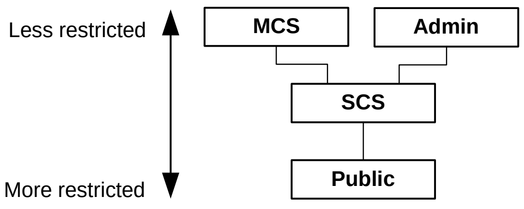

Figure 4 Hierarchical structure of groups

The structure is divided into three levels, where a higher level means fewer restrictions. The two groups MCS and Admin are placed at the highest level. The tasks provided to each of these groups are not accessible from the other groups.

Some tasks may require a limitation in the number of users active in that task. An example is management of the flight plan, which is limited to one active user from the MCS group. This is implemented in the group validation procedure.

A user may not be active in more than one session at a time, because of the limits in some tasks. This is implemented in the user validation procedure to avoid situations in which users forget to log off and thus block for new logons.

In connection with this, improper logouts of the system is a problem. This might be caused by the user when moving to another web page without performing a logout or by web browser crashes. This problem may cause groups to be locked infinitely or disallow user logon.

Group Validation

A script handles control of restrictions. This also specifies the maximum number of active members of each group.

To perform group validation, a procedure is implemented as depicted in figure 5.

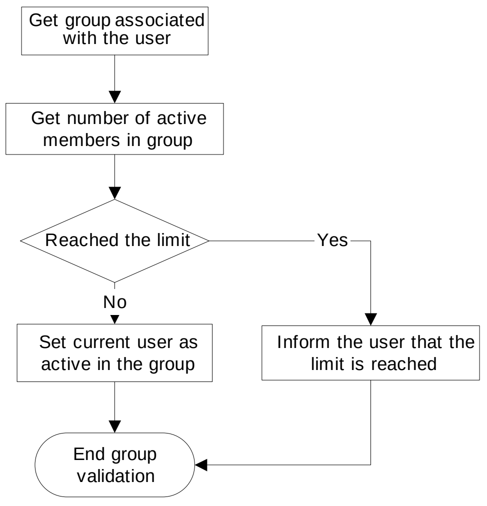
Get group associated with the user
Get number of active members in group
Reached the limit
Yes
No
Set current user as active in the group
Inform the user that the limit is reached
End group validation

Figure 5 Flowchart of group validation

First, the group associated with the user is determined. Next step is to check the number of active users currently in the group and compare it with the limit. If the limit is reached, the user is denied access and informed of this. Otherwise, the user is granted access to the group.

User Validation

To be able to perform user validation each web browser session is provided with a unique identification (id). The id is generated by PHP every time a user enters the distributed control room with a browser. A user logging on to the system is associated with this id. The logon procedure is depicted in Figure 6.

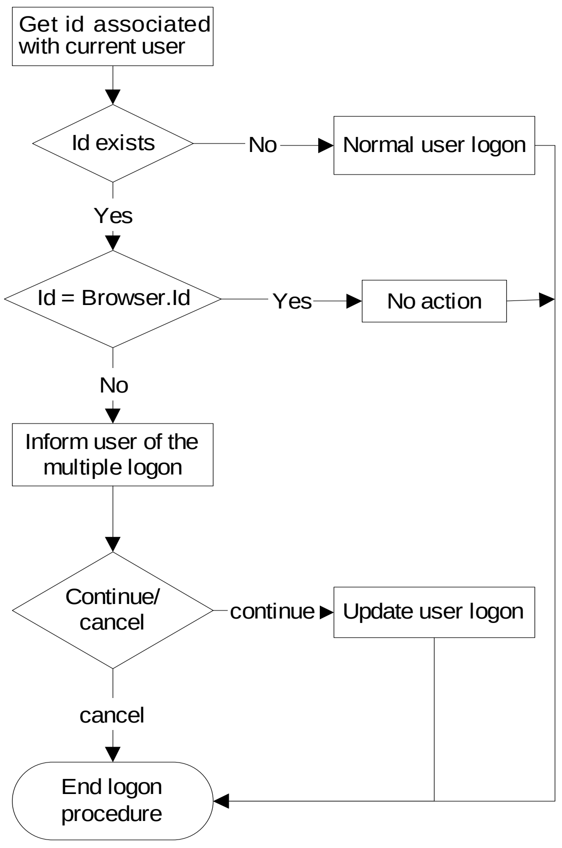

Figure 6 Flow diagram of the user logon procedure

First, the system checks for any existing id associated with the user. If no id exists, the normal logon procedure is executed. Otherwise, the id is compared with the current session id of the web browser. If the id's are identical, the user tries to perform a multiple logon with the same web browser and no action is taken. If the id's are not identical, the user probably has forgotten to log out from one workstation and is trying to logon to another. In this situation, the user is given two choices: continue the logon or cancel the logon. When continuing the logon, the existing session is closed and the new session id of the web browser is stored in the database. If the user chooses to cancel, the login procedure registers the action as a public user login.

System Logout

In order to handle improper logouts a timeout of each user logon is implemented. This timeout is reset each time the user performs an action in the system. In case of inactivity, the user is automatically logged out from the system. This method prevents lock up of the system regarding user limitation.

Secure Communication

Information in transit over the Internet may be intercepted and modified by unauthorized persons. To protect sensitive information the communication is encrypted by using the Secure Socket Layer (SSL) protocol [SSL], which is supported by common web browsers and web servers.

As everybody has access to the public areas, there is no need to protect these, unlike all other areas, which has to be secured. This differentiating between protected and unprotected areas is achieved by using a table in a database. Every time a page is requested, the system checks the need for secure communication. Then the system checks whether the communication is already secure or not. This is illustrated in figure 7.

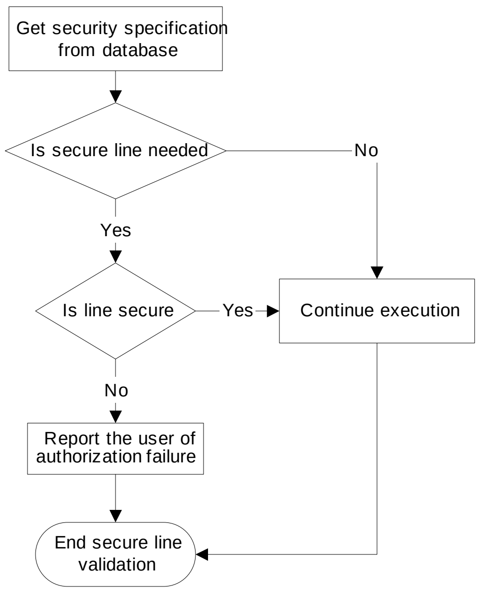
Get security specification from database
Is secure line needed
No
Yes
Is line secure
Yes
Continue execution
No
Report the user of authorization failure
End secure line validation

Figure 7 Flowchart of secure communication validation

Validation and Test

Each time a script is executed a validation process is needed. To ease the implementation and testing, a centralized script combines the validation procedures, joining the use case validation, user validation and the secure communication validation in one table. In this way, less database queries are needed to carry out the validations.

To verify that the validations are performed correctly a log is recorded in the database. The log records all events in the system and therefore contains enough information to reconstruct any scenario. This provides information to help locating errors.

Carrying out a test of the validation system requires tests of the three security topics dealt with until now. The tests performed are extensive but deals with only one topic at a time. The three validations are tested by specifying a number of interaction scenarios and the expected result. After going through a scenario, the log is compared to the expected result to ensure that the validation system is functioning correctly.

Test Results

The test verifies that the use case validation system logs state transitions and returns an error to the user if an illegal state transition is attempted.

The access control is capable of granting access to the system sections, which a user has, access to, and deny access to the rest of the system. The access control also blocks for concurrent access from more than one MCS. If a MCS tries to logon and one MCS already is logged on the latest arrived is informed and logged on as SCS. Finally, the access control disallows multiple logons from one user.

Secure communication is achieved to the parts of the system that require this. The secure communication is switched on as the user moves to the parts of the system, which need to be protected.

Problems

The system operates like intended but there are limitations due to the way the system is designed.

The first of these limitations is that a person only is allowed to be a member of one user group. Often a person needs to be both an administrator and a MCS. The other limitation is that only one MCS may be logged on at a time.

In a small-scale system such as this small-satellite control room, this is not a problem, but it might be in larger systems. Both limitations can be eliminated by a redesign of the system but it will increase the complexity.

Another problem related to the access control system is improper logouts. An example of such a problem is if a user closes the web browser without logging of the system. This problem was solved by the use of timeouts on inactive users.

The use case validation system guarantees that the user's interaction with the system is correct according to the use cases stored in the database.

This system registers every action and state transition in a log. The log was a valuable tool to locate errors in the system throughout the test phase of the development. After release, it can be used to monitor the system.

The system works as intended. It registers if a user does not follow the designed use cases when navigating around in the control room.

However, a problem was encountered when reporting events in the log: The size of the log grows fast and it gets harder to extract the needed information. Therefore, the types of information in the log should be carefully chosen when the system is released.

Another problem regarding the use case validating system is the complexity of the web application. With a complex application, the state transactions table will grow and it will be difficult to maintain. To solve this, a system that simplifies management of the state transition table is needed.

The problems encountered are general even though a specific analysis and design of a distributed control room for a small-satellite forms the basis for the research. It is concluded that the technologies chosen, are capable of offering a security level, which is acceptable in many scientific and industrial applications.

References

[JSWITCH] **JSWITCH/JSAT: REAL-TIME AND OFFLINE WORLD WIDE WEB INTERFACE**
http://isc.gsfc.nasa.gov/Papers/DOC/JSWITCH.PDF, December 2001

[PHP] **PHP homepage**
http://www.php.net, December 2001

[SSL] **Apache and Secure Transactions**
http://www.apacheweek.com/features/ssl, December 2001

[OOAD] **Objekt Orienteret Analyse og Design**
Lars Mathiasen, Andreas Munk Madsen, Peter Axel Nielsen, Jan Stage
2nd edition, 1998
ISBN: 87-7751-129-8