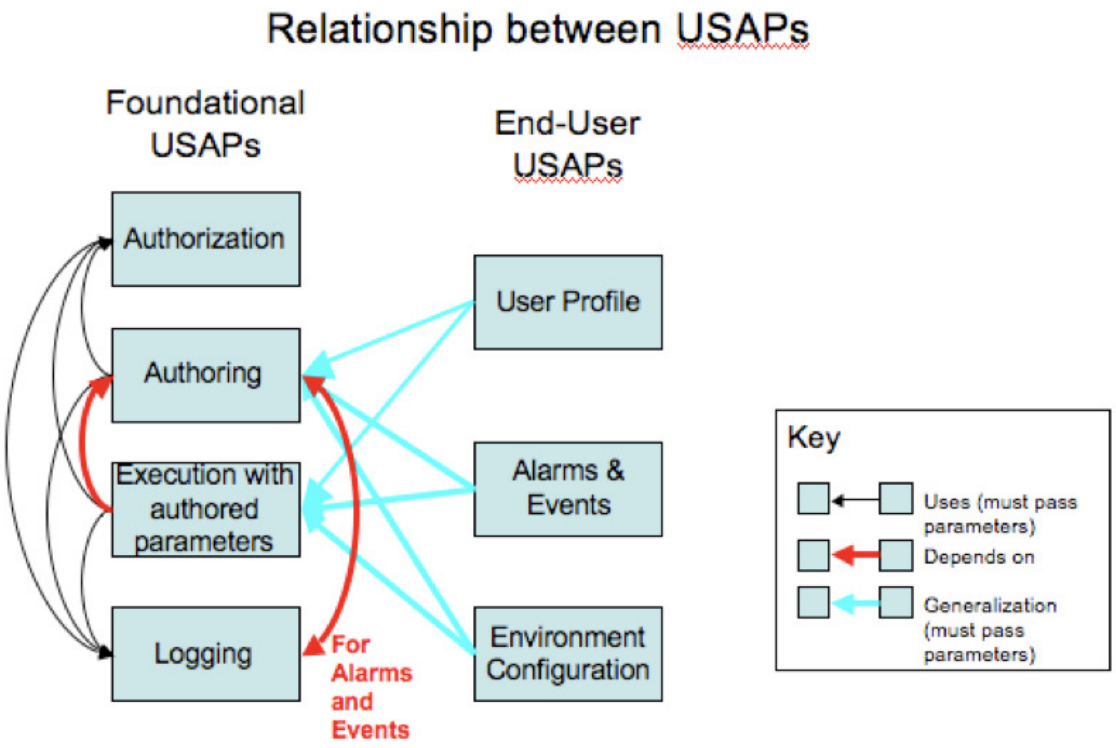
Pattern Language for Usability Supporting Architectural Patterns

Bonnie E. John, Len Bass, Elspeth Golden
28July08

Assumptions about USAPs

- There are intra-responsibility assumptions (e.g., Communication between involved portions of the system is assumed).
- There are inter-responsibility, intra-USAP assumptions (e.g., things done by one responsibility are accessible to other responsibilities in the USAP)
- There are inter-USAP assumptions (e.g., that the protocol for saving an authored specification is defined and known to the developers of both the authoring system and the execution system). These are "in" the red arrows in the diagram below and we just put in a "shared assumptions" section into the front-matter of each USAP.
- There are assumptions about the execution environment for systems informed by USAPs (i.e., code developed unconnected to USAPs and code developed by considering USAPs). For example, USAPs assume that there is a portion of the system that displays information to a user.



1 Foundational USAPs

For each Foundational USAP, provide:

Purpose:

- A general statement of the purpose of the foundational USAP.

Justification:

- Justification for this foundational USAP

Glossary:

- Definitions of terms introduced in this foundation USAP

Parameters needed by this USAP:

- These parameters are used to make the foundational USAP specific to referring USAPs.

(NOTE: Our expectation is that the following information will ultimately be generated automatically from the USAPs that use and are used by this USAP either directly or indirectly, otherwise there will be a maintenance headache.)

For each Foundational USAP used by this USAP (if any), provide:

Parameters furnished to the foundational USAP:

- The parameter values furnished to the foundational USAP used by this USAP.

For each End-User USAP that refers to this foundational USAP, provide

End-User USAP interpretation:

- The parameter values furnished to this foundational USAP by the referring end-user USAP.

1.1 Authorization

Purpose:

The Authorization Foundational USAP's purpose is to identify and authenticate users (human or other systems) of the system.

Justification:

Users must be authorized when security or personalization is important.

Glossary:

- To be completed after user testing

Parameters needed by this USAP:

- USER: Who the users are, i.e., the role they play. These can be human and/or other systems.
- ACTIVITY: What activities authorization will authorize (i.e., permissions).

Foundational USAPs used by this USAP:

None.

USAPs that use this Foundational USAP:

In the User Profile End-User USAP, there are two paths to this USAP, with two sets of parameters.
Through the Authoring Foundational USAP:

- USER: Author.
  ACTIVITY: Author user profile.

Through the Execution with Authored Parameters Foundational USAP:

- USER: End user.
  ACTIVITY: Execute the system with parameter values from user profile.

In the Environment Configuration End-User USAP, there are two paths to this USAP, with two sets of parameters.
Through the Authoring Foundational USAP:

- USER: Author.
  ACTIVITY: Author configuration description.

If necessary environment information must be entered by an authorized user, then through the Execution with Authored Parameters Foundational USAP:

- USER: End user.
  ACTIVITY: Execute the system with parameter values from configuration description.

In the Alarms and Events End-User USAP, there are two paths to this USAP.
Through the Authoring Foundational USAP:

- USER: Author.
  ACTIVITY: Author alarm and event rules and displays.

Through the Execution with Authored Parameters Foundational USAP:

- USER: End user.
  ACTIVITY: Author alarm and event rules and displays.

1.1.1 Identification

1.1.1.1 The system has to provide a way for the USER to input his, her or its identity. (For a human user, this input could be user typing a login ID, running a finger over a fingerprint reader, or standing in front of a camera for face recognition, etc. For a system user, this input could be an IP address, a previously specified identity, etc.)

1.1.1.1.1.1.1 Rationale

Software has no way of recognizing a USER without explicit input.
USERs want to access the system.
The environment contains multiple potential users only some of whom are allowed to use the system.

1.1.1.1.1.1.2 Implementing this responsibility

The USERs are involved because they must take explicit action.

The portion of the system that receives user input is involved, because users' explicit action must be handled by the system.
There must be a portion of the system that processes the input.

1.1.1.2 The system should inform the USER and/or the system administrator of the results of the identification. Typically, a successful identification is indicated by allowing the USER to proceed. (If logging of identification results is desired see Logging Foundational USAP with parameter CONTEXT=Identification.)

1.1.1.2.1.1.1 Rationale

USERs want to know if their identification succeeded so they can proceed.
USERs want to know if their identification failed because they cannot proceed without it.
System administrators might want to know of a failed identification because it might be an indication of unauthorized users attempting to access the system.
The environment contains multiple potential users, some of whom might be malicious.

1.1.1.2.1.1.2 Implementing this responsibility

In the event of failing to be identified,

The portion of the system that does the identification must provide information about the failure.

If informing the system administrator:

In the event of a failure, a portion of the system must have a mechanism to inform the system administrator. This may be quite complicated (e.g., sending email) and is beyond the scope of this USAP.

For informing the USER:

In the event of a failure, the portion of the system that renders information to the user should display information about the failure. Often this feedback will wait for authentication information to be input and the feedback will be of the form that this userID/password are not known to the system.

In the event of a successful identification, the portion of the system that renders information to the user should indicate success in some way (e.g., simply letting the USER proceed).

The portion of the system that handles input from the user should provide a UI to dismiss any unwanted information.

1.1.1.3 The system must remember the USER's identity for the duration of the session.

1.1.1.3.1.1.1 Rationale

The software will use this information later (e.g., in determining authorization).
The USER only wants to enter this information once per session.
The environment contains multiple potential users only some of whom are allowed to use the system.

1.1.1.3.1.1.2 Implementing this responsibility

There must be a portion of the system that maintains the USER's identity.

1.1.1.4 The system must display the identity of the current USER.

1.1.1.4.1.1.1 Rationale

The USER wants to be sure that the system knows who he or she is.

1.1.1.4.1.1.2 Implementing this responsibility

The portion of the system that maintains the USER's identity provides the USER's identity.
The portion of the system that renders information to the user displays the USER's identity.
The portion of the system that handles input from the user should provide a UI to dismiss any unwanted information.

1.1.1.5 The system must provide a way for the USER to recover from mistakes in identification. (This could be an erroneous input by the user or the system or it could be the user forgetting his or her user ID. In the case of erroneous input, the solution could be as simple as re-entering the information, so this section will expand on the case of forgetting.)

1.1.1.5.1.1.1 Rationale

USERs sometimes forget the way to identify themselves to the system.
Software has no way of recognizing a USER without correct explicit input.
The environment contains multiple potential users only some of whom are allowed to use the system.

1.1.1.5.1.1.2 Implementing this responsibility

The USERs are involved because they must explicitly indicate that they forgot their ID.
The portion of the system that receives user input is involved, because users' explicit action must be handled by the system.
There must be a portion of the system that has a mechanism to inform the USER of the correct ID. This mechanism may be quite complicated (e.g., sending email to a previously-stored address), and is beyond the scope of this USAP.

1.1.2 Authentication

1.1.2.1 The system has to provide a way for the USER to input his, her or its authentication. (For a human user , this input could be user typing a password, running a finger over a fingerprint reader, or standing in front of a camera for face recognition, etc. For a system user, this input could be a certificate or previous authentication, etc.)

1.1.2.1.1.1.1 Rationale

Software has no way of authenticating a USER without explicit input.
USERs want to know that they are the only ones able to take action under their names.
The environment contains multiple potential users only some of whom are allowed to use the system.
The organization wants to restrict access to authorized USERs and to have USERs be accountable for their actions.

1.1.2.1.1.1.2 Implementing this responsibility

The USERs are involved because they must take explicit action.
The portion of the system that receives user input is involved, because users' explicit action must be handled by the system.
There must be a portion of the system that processes the input.

1.1.2.2 The system must perform and remember the results of the authentication. (This could be matching a password, recognizing a fingerprint, face recognition, etc. This might be a multi-stage process, e.g., first validating a user's ID and then matching it to the password.)

1.1.2.2.1.1.1 Rationale

The system is the entity that has the information necessary to perform the authentication.

1.1.2.2.1.1.2 Implementing this responsibility

The portion of the system that maintains the USER's identity provides the USER's identity.
There must be a portion of the system that performs the authentication.
The portion of the system that maintains the USER's identity must remember whether this USER has been authenticated.

1.1.2.3 The system should inform the USER and/or the system administrator of the results of the authentication. (If logging of identification results is desired see Logging Foundational USAP with parameter CONTEXT=Authentication.)

1.1.2.3.1.1.1 Rationale

USERs want to know if their authentication succeeded so they can proceed.

USERs want to know if their authentication failed because they cannot proceed without it.
System administrators might want to know of a successful authentication for audit trail purposes.
System administrators might want to know of a failed authentication because it might be an indication of unauthorized users attempting to access the system.
The environment contains multiple potential users, some of whom might be malicious.

1.1.2.3.1.1.2 Implementing this responsibility

In the event of failing to be authenticated,
The portion of the system that does the authentication must provide information about the failure.
If informing the system administrator:
In any event, a portion of the system must have a mechanism to inform the system administrator. This may be quite complicated (e.g., sending email) and is beyond the scope of this USAP.
For informing the USER:
In the event of a failure, the portion of the system that renders information to the user should display information about the failure.
In the event of a successful authentication, the portion of the system that renders information to the user should indicate success in some way (e.g., a welcome message or simply letting the user proceed).
The portion of the system that handles input from the user should provide a UI to dismiss any unwanted information.

1.1.3 Permissions

1.1.3.1 The system must permit or prohibit specific ACTIVITIES dependent on who the USER is and on the set of ACTIVITIES they are attempting to perform, i.e., the system must know (built in, stored, provided by the OS, etc.) the specific permissions for each USER and then enforce these permissions

1.1.3.1.1.1.1 Rationale

The organization may wish to allow different USERs to have different capabilities.
The USERs want access to the ACTIVITIES they have permission for.
The system has to have a concept of permissions to be able to allow or disallow ACTIVITIES.
The system has to know the mapping between USERs and ACTIVITIES to grant different permissions to different users.

1.1.3.1.1.1.2 Implementing this responsibility

The portion of the system that maintains the USER's authenticated identity provides the USER's identity.
The portion of the system that does the ACTIVITIES must provide the requested ACTIVITIES.

There must be a portion of the system that maintains the mapping between USERs and their permitted ACTIVITIES.
This latter portion of the system must use the USER's authenticated identity, the particular ACTIVITIES desired and the map to determine whether the ACTIVITIES are allowed.

1.1.3.2 The USER and/or the system administrator should be informed when permission is granted or denied for doing ACTIVITIES. Typically, this is done through the portion of the system that requests the ACTIVITIES.

1.1.3.2.1.1.1 Rationale

USERs want to know if their operations failed because of permission.
System administrators might want to know if permission is denied because it might be an indication of users attempting to exceed their permissions.
The environment contains multiple potential users, some of whom might be malicious.

1.1.3.2.1.1.2 Implementing this responsibility

In the event of failing to be allowed to do ACTIVITIES,
The portion of the system that does the ACTIVITIES must provide information about the failure.
If informing the system administrator:
A portion of the system must have a mechanism to inform the system administrator. This may be quite complicated (e.g., sending email) and is beyond the scope of this USAP.
For informing the USER:
In the event of a failure, the portion of the system that renders information to the user should display information about the failure.
In the event of a successful permission, the portion of the system that does the ACTIVITIES should proceed and inform the USER appropriately, typically through the results of performing the ACTIVITIES.
The portion of the system that handles input from the user should provide a UI to dismiss any unwanted information.

1.1.4 **The system must have a way for the USER to log off.** (If there are requests still pending (e.g., unsaved changes), then notify the user and ask for confirmation of the log-off request. Consider some way to protect the system if the confirmation if not forthcoming in a reasonable amount of time.)

1.1.4.1.1.1.1 Rationale

Software has no way of knowing that a USER is finished without explicit input.
USERs wants to protect their work.

The environment contains multiple potential users, some of whom may be malicious, so logging off prevents them from having access.

1.1.4.1.1.1.2 Implementing this responsibility

If log-off is at the USER's request:
The system must provide a means for the USER to indicate a desire to log-off (e.g., a button, keyboard shortcut, voice command).
The USER must indicate their desire to log off and respond to any confirmation request.
The portions of the system that do the ACTIVITIES must ask for confirmation in the event of request still pending. This involves keeping track of pending requests and requesting the appropriate interactions with the USER.
There must be a portion of the system that de-authenticates the USER after receiving the log-off request and any necessary confirmations.

If with a system-initiated request:
There must be a portion of the system with a mechanism to initiate a log-off request. (This may be as simple as a time-out or quite complicated (e.g., shutting down the entire system), and is beyond the scope of this USAP.)
If there are pending requests, waiting for the USER to confirm is not a good idea, either the requests are aborted or the log-off is aborted.

1.2 Authoring

Purpose:
The Authoring Foundational USAP's purpose is to allow specification of the behavior of the system in certain ways under certain circumstances.

Justification:
Users want to control the behavior of the system in certain ways under certain circumstances without having to set it up every time. The system needs a specification of parameters to determine its behavior in these circumstances. Therefore the user must author a specification of parameters that will subsequently be used upon execution (see Foundational USAP Execution with Authored Parameters).

Glossary:
- To be completed after user testing

Parameters needed by this USAP:
- SPECIFICATION: The persistent parameter values that are authored.
- APPROPRIATENESS-INFORMATION: The circumstances under which it is appropriate to use a particular SPECIFICATION.

Shared Assumptions:
Shared with any other USAP that uses the SPECIFICATION:
- The syntax and semantics for the concepts that are included in the SPECIFICATION are defined and known to the developers of both the authoring system and the systems informed by other USAPs that share the SPECIFICATION.

- The protocol for saving the SPECIFICATION is defined and known to the developers of both the authoring system and the systems informed by other USAPs that share the SPECIFICATION.
- USAPs sharing these assumptions
    - Execution with Authored Parameters foundational USAP

Foundational USAPs used by this USAP:
Authorization Foundational USAP:
- USER: Author
- ACTIVITY: Author the SPECIFICATION

USAPs that use this Foundational USAP:

(Optional) Logging Foundational USAP
- SPECIFICATION: Logging specification.
- APPROPRIATENESS-INFORMATION: Logging state.
-

(Optional) Execution with Authored Parameters USAP
- SPECIFICATION: Execution with Authored Parameters.SPECIFICATION
  (this is the SPECIFICATION parameter that was passed to Execution with Authored Parameters)
- APPROPRIATENESS-INFORMATION: Execution with Authored Parameters. APPROPRIATENESS-INFORMATION
  (this is the APPROPRIATENESS-INFORMATION parameter that was passed to Execution with Authored Parameters)

User Profile End-user USAP:
- SPECIFICATION: User profile.
- APPROPRIATENESS-INFORMATION: User identity.

Environment Configuration End-User USAP:
- SPECIFICATION: Configuration description.
- APPROPRIATENESS-INFORMATION: Environment identity.

Alarms and Events End-User USAP:
- SPECIFICATION: Rules for Alarms, Events and Displays.
- APPROPRIATENESS-INFORMATION: Context of use.

1.2.1 Create a SPECIFICATION

1.2.1.1 The system must provide a way for an authorized author to create a SPECIFICATION. (See Authorization Foundational USAP with parameters USER=Author and ACTIVITY=Author the SPECIFICATION.)

1.2.1.1.1.1.1 Rationale

A SPECIFICATION doesn't exist unless it is created.

1.2.1.1.1.1.2 Implementing this responsibility

The portion of the system that renders output must render a UI that allows the parameters to be specified and displays existing values.
The portion of the system that accepts input from the user must accept parameters.
There must be a portion of the system with a mechanism to create new SPECIFICATIONs.

1.2.1.2 Consider providing default values for all specifiable parameters when a SPECIFICATION is created. (Providing defaults simplifies the creation process, but may increase the probability of error for environments that deviate from those defaults. If the cost of error is high, then consider not providing defaults or requiring confirmation of each value. Default values can be changed through modification.)

1.2.1.2.1.1.1 Rationale

System has to have *something* so defaults might be provided.
Circumstances might be very similar so defaults might capture that similarity.
Authors may want to be efficient and defaults may save authoring time.
Authors may only be interested in changing specific aspects of the SPECIFICATION, so having defaults for the rest of it is useful.

1.2.1.2.1.1.2 Implementing this responsibility

The portion of the system that creates a new SPECIFICATION must assign defaults.

1.2.1.3 The SPECIFICATION must be given an identifier. The identifier should be treated as a parameter that has a default value and can be modified by the user. (One mechanism for assigning this identification might be a "Save as".)

1.2.1.3.1.1.1 Rationale

Circumstances might be very similar so values in one SPECIFICATION may transfer to other circumstances.
Authors may want to be efficient and may want to begin the specification process with a similar SPECIFICATION.
The authoring system must have an identifier to distinguish one SPECIFICATION from another.

1.2.1.3.1.1.2 Implementing this responsibility

<u>If the identifier is provided by a author:</u>
The portion of the system that renders output must render a UI that allows the author to provide an identifier and display it.
The portion of the system that accepts input from the user must accept the identifier.

There must be a portion of the system that manages the authoring process.
The portion of the system that manages the authoring process must associate the SPECIFICATION with the identifier.

If the identifier is provided automatically by the system:
The portion of the system that creates the SPECIFICATION generates an identifier (e.g., MSWord automatically generates a name for a new document).
The portion of the system that manages the authoring process must associate the SPECIFICATION with the identifier.

1.2.1.4 The SPECIFICATION must be associated with the APPROPRIATENESS-INFORMATION (i.e., the circumstances under which it should be invoked (e.g., for specific users, roles, environments)). The APPROPRIATENESS-INFORMATION should be treated as a parameter that has a default value and can be modified by the author.

1.2.1.4.1.1.1 Rationale

The SPECIFICATION is only appropriate for use under certain circumstances.
System can only use a SPECIFICATION if it is associated with a circumstance.
Users want the system to work for them the way they want it to work when they want it to work that way.

1.2.1.4.1.1.2 Implementing this responsibility

If the association is provided by an author:
The portion of the system that renders output must render a UI that allows the authoring of APPROPRIATENESS-INFORMATION and display existing values.
The portion of the system that accepts input from the user must accept the APPROPRIATENESS-INFORMATION.
The portion of the system that manages the authoring process must associate the SPECIFICATION with the APPROPRIATENESS-INFORMATION.

If the association is provided automatically by the system when a new user or role is created:
The portion of the system that creates the SPECIFICATION generates an association.
The portion of the system that manages the authoring process associates the SPECIFICATION with the circumstances.

1.2.2 **Save a SPECIFICATION.** The system must provide a means for an authorized author to save and/or export the SPECIFICATION (e.g., by autosave or by author request). (See Authorization Foundational USAP with parameters USER=Author and ACTIVITY=Author the SPECIFICATION.) If other systems are going to use the SPECIFICATION, then use a format that can be used by the other

systems. (If logging of authoring results is desired see Logging Foundational USAP with parameter CONTEXT=Authoring.)

1.2.2.1.1.1.1 Rationale

Authors want to be efficient (i.e., input information into the SPECIFICATION only once).
The system can only remember things if they are persistent from session to session.
The software may need to share a SPECIFICATION with other software.

1.2.2.1.1.1.2 Implementing this responsibility

If the initiation of the save was automatic:
That portion of the system that manages the authoring process performs the initiation.
That portion of the system that manages the authoring process stores and/or exports the SPECIFICATION.
If the initiation of the save was at the author's request:
The portion of the system that renders output must render a UI that allows the parameters needed by the system (e.g., format, location) to be input and display them.
The portion of the system that accepts input from the user must accept the parameters.
That portion of the system that manages the authoring process stores and/or exports the SPECIFICATION.

1.2.3 Modify a SPECIFICATION

1.2.3.1 Provide a way for an authorized author to retrieve a SPECIFICATION (e.g., import a previously-saved file, utilize a previously-generated data structure, or restore to default values). (See Authorization Foundational USAP with parameters USER=Author and ACTIVITY=Author the SPECIFICATION.)

1.2.3.1.1.1.1 Rationale

Authors might want different values than are currently assigned.
System has stored the information and must have a current set of data to work with.

1.2.3.1.1.1.2 Implementing this responsibility

The portion of the system that renders output must render a UI that allows the author to request a retrieval of a SPECIFICATION.
The portion of the system that accepts input from the user must accept this request.
That portion of the system that manages the authoring process retrieves the SPECIFICATION.

1.2.3.2 Display the current parameter values (including identifier and circumstances).

1.2.3.2.1.1.1 Rationale

Authors want to see what they are editing.

1.2.3.2.1.1.2 Implementing this responsibility

That portion of the system that manages the authoring process must provide the values.
The portion of the system that renders output must render a UI that displays the values.
The portion of the system that handles input from the user should provide a UI to dismiss any unwanted information.

1.2.3.3 The system must provide a ways for an authorized author to change the parameter values. The syntax and semantics of the values specified should conform to the assumptions of the execution environment of the system. (More details: Best practice is to constrain the author to such conformation (e.g., choose from drop-down list, provide a slider for a range of values). If the author's choices are not constrained, then the system should check the syntax and semantics and provide feedback if either are unreasonable.) (See Authorization Foundational USAP with parameters USER=Author and ACTIVITY=Author the SPECIFICATION.)

1.2.3.3.1.1.1 Rationale

Authors might want different values than are currently assigned.

1.2.3.3.1.1.2 Implementing this responsibility

The portion of the system that renders output must render a UI that allows values to be changed.
The portion of the system that accepts input from the user must accept new values.
That portion of the system that manages the authroing process replaces the current values with the new values.

1.2.4 Delete a SPECIFICATION

1.2.4.1 Provide a way for an authorized author to tentatively remove a SPECIFICATION from the system (e.g., analogous to dragging a file to the trash). The system might require a confirmation from the author prior to performing this action. (See Authorization Foundational USAP with parameters USER=Author and ACTIVITY=Author the SPECIFICATION.) (If logging of authoring deletions is desired see Logging Foundational USAP with parameter CONTEXT=Authoring.)

1.2.4.1.1.1.1 Rationale

Authors might accidentally delete a SPECIFICATION and want to restore it.
The organization doesn't want extraneous SPECIFICATIONs on the system.
The system has to keep it around in case it has to be restored.

1.2.4.1.1.1.2 Implementing this responsibility

The portion of the system that renders output must render a UI that allows the author to tentatively delete a SPECIFICATION.
The portion of the system that accepts input from the user must accept this request.
That portion of the system that manages the authoring process saves the SPECIFICATION in case has to be restored.
The portion of the system that renders output must indicate that the SPECIFICATION has been (tentatively) deleted.

1.2.4.2 Retrieve a tentatively-deleted SPECIFICATION from the system (e.g., analogous to dragging a file out of the trash)

1.2.4.2.1.1.1 Rationale

Authors might want to restore an accidentally deleted SPECIFICATION.
The system has kept it around so that it can be restored.

1.2.4.2.1.1.2 Implementing this responsibility

The portion of the system that renders output must render a UI that allows the author to restore a tentatively deleted SPECIFICATION.
The portion of the system that accepts input from the user must accept this request.
That portion of the system that manages the authoring process must restore the SPECIFICATION.
The portion of the system that renders output must indicate that the SPECIFICATION has been restored.

1.2.4.3 Provide a way for an authorized author to permanently remove a SPECIFICATION from the system (e.g., analogous to emptying the trash). The system should require a confirmation from the author prior to performing this action. (See Authorization Foundational USAP with parameters USER=Author and ACTIVITY=Author the SPECIFICATION.)

1.2.4.3.1.1.1 Rationale

Authors want to delete a SPECIFICATION that is no longer relevant to their needs.
The organization doesn't want extraneous SPECIFICATIONs on the system.
The system has limited resources.

1.2.4.3.1.1.2 Implementing this responsibility

The portion of the system that renders output must render a UI that allows the author to permanently delete a SPECIFICATION.
The portion of the system that accepts input from the user must accept this request.
That portion of the system that manages the authoring process permanently deletes the SPECIFICATION.
The portion of the system that renders output must indicate that the SPECIFICATION has been deleted.

1.2.5 The system must have a way for the author to exit the authoring system. (If there are requests still pending (e.g., unsaved changes), then notify the author and ask for confirmation of the exit request.)

1.2.5.1.1.1.1 Rationale

Software has no way of knowing that an author is finished without explicit input. Authors want to exit when they are done.

1.2.5.1.1.1.2 Implementing this responsibility

The system must provide a means for the author to indicate a desire to exit (e.g., a button, keyboard shortcut, voice command).
The authors must indicate their desire to exit and respond to any confirmation request.
The portions of the system that do the authoring activities must ask for confirmation in the event of request still pending. This involves keeping track of pending requests and requesting the appropriate interactions with the author.

1.3 Execution with Authored Parameters

Purpose:

- The Execution with Authored Parameters USAP's purpose is to allow a system to use a specification of parameters to determine its behavior in the areas in which the parameters apply.

Justification:

- Users want to control the behavior of the computer in certain ways under certain circumstances that they have previously specified (see the Authoring foundational USAP).

Glossary:

- To be completed after user testing

Parameters needed by this USAP:

- SPECIFICATION: The persistent parameter values that have been previously specified.
- APPROPRIATENESS-INFORMATION: The information necessary to locate the appropriate SPECIFICATION.

Shared Assumptions:
Shared with any other USAP that uses the SPECIFICATION:

- The syntax and semantics for the concepts that are included in the SPECIFICATION are defined and known to the developers of both the authoring system and the systems informed by other USAPs that share the SPECIFICATION.
- The protocol for saving is defined and known to the developers of both the authoring system and the systems informed by the Execution with Authored Parameters USAP.
- USAPs sharing this assumption
    - Authoring Foundational USAP

Foundational USAPs used by this USAP:

Authorization Foundational USAP:
- USER: End user
- ACTIVITY: Execute the system with parameter values from SPECIFICATION

(Optionally) Authoring Foundational USAP:
- SPECIFICATION: SPECIFICATION.
- APPROPRIATENESS-INFORMATION: APPROPRIATENESS-INFORMATION.

USAPs that use this Foundational USAP:
User Profile End-user USAP:
- SPECIFICATION: User profile.
- APPROPRIATENESS-INFORMATION: User identity

Environment Configuration End-User USAP:
- SPECIFICATION: Configuration description.
- APPROPRIATENESS-INFORMATION: Environment identity

Alarms and Events End-User USAP:
- SPECIFICATION: Rules for Alarms, Events and Displays.
- APPROPRIATENESS-INFORMATION: Context of use

1.3.1 Access the appropriate SPECIFICATION

1.3.1.1 Retrieve APPROPRIATENESS-INFORMATION (which will allow determination of appropriate SPECIFICATION). (If APPROPRIATENESS-INFORMATION depends on the user then the user must be authorized. See Authorization USAP with parameters USER=End user and ACTIVITY=Execute the system with parameter values from SPECIFICATION.)

1.3.1.1.1.1.1 Rationale

The system needs a SPECIFICATION in order to execute appropriately.
System has no way to determine appropriateness of a SPECIFICATION without APPROPRIATENESS-INFORMATION. Sometimes this information may have to come from a user, sometimes it can be inferred from the environment.
Users and organizations want the system to execute appropriately.

1.3.1.1.1.1.2 Implementing this responsibility

There must be a portion of the system that knows how to retrieve APPROPRIATENESS-INFORMATION. For example, APPROPRIATENESS-INFORMATION may be maintained in a fixed location within a system or within the file structure of the system.

1.3.1.2 The system must retrieve the appropriate SPECIFICATION.

1.3.1.2.1.1.1 Rationale

The system needs a SPECIFICATION in order to execute appropriately.
Users and organizations want the system to execute appropriately.

1.3.1.2.1.1.2 Implementing this responsibility

The portion of the system that will use the specified parameters must retrieve the SPECIFICATION.

1.3.1.3 The system should inform the user and/or the system administrator of the results of attempting to retrieve the appropriate SPECIFICATION. There are three cases: find zero, find one, find many. (Typically, finding zero generates an error message, finding one is indicated by allowing the user to proceed, finding many is indicated by listing their identifiers and allowing the user to view the contents of the SPECIFICATION.)

1.3.1.3.1.1.1 Rationale

Users want to know if the appropriate SPECIFICATION has been located so they can proceed.
Users want to know if the appropriate SPECIFICATION has not been located because they cannot proceed without it.
Users want to know if more than one appropriate SPECIFICATION has been located because they can help resolve the ambiguity.
System administrators might want to know if the appropriate SPECIFICATION has not been located or if there are many, because it might be an indication of system error.

1.3.1.3.1.1.2 Implementing this responsibility

In the event of failing to be located or finding more than one appropriate SPECIFICATION:
The portion of the system that does the locating must provide information about the failure or the identities of each SPECIFICATION.
If informing the system administrator:
In the event of a failure or locating more than one appropriate SPECIFICATION, a portion of the system must have a mechanism to inform the system administrator. This may be quite complicated (e.g., sending email) and is beyond the scope of this USAP.
For informing the user:
In the event of a failure:
The portion of the system that renders information to the user should display information about the failure.
In the event of a successfully locating an appropriate SPECIFICATION:
The portion of the system that renders information to the user displays should

indicate success in some way (e.g., progress feedback saying it is retrieving the SPECIFICATION).

In the event of finding many:

The portion of the system that renders information to the user should indicate the identifier of each SPECIFICATION.

The portion of the system that renders information to the user should provide a UI for allowing the user to resolve the ambiguity.

The portion of the system that handles input from the user should allow the user to select one of the located SPECIFICATIONs or request to view the contents of one or more of the located SPECIFICATIONs.

The portion of the system that handles input from the user should provide a UI to dismiss any unwanted information.

1.3.1.4 The system must check that the retrieved SPECIFICATION is valid for use.

1.3.1.4.1.1.1 Rationale

System cannot use an invalid SPECIFICATION.

1.3.1.4.1.1.2 Implementing this responsibility

The portion of the system that will use the specified parameters must check the SPECIFICATION to make sure it is valid (e.g., it could be empty, corrupt, or incomplete).

1.3.1.5 The system should inform the user and/or the system administrator of the results of validating the SPECIFICATION.

1.3.1.5.1.1.1 Rationale

Users want to know if the SPECIFICATION is valid so they can proceed.
Users want to know if the SPECIFICATION is not valid because they cannot proceed with an invalid SPECIFICATION.
System administrators might want to know if the SPECIFICATION is not valid, because it might be an indication of system error.

1.3.1.5.1.1.2 Implementing this responsibility

In the event of an invalid SPECIFICATION:

The portion of the system that does the validation must provide information about the failure.

If informing the system administrator:

In the event of a failure, a portion of the system must have a mechanism to inform the system administrator. This may be quite complicated (e.g., sending email) and is beyond the scope of this USAP.

For informing the user:

In the event of an invalid SPECIFICATION:

The portion of the system that renders information to the user should display information about the failure.

In the event of a valid SPECIFICATION:

The portion of the system that renders information to the user displays should indicate success in some way. This is typically indicated by allowing the user to proceed.

The portion of the system that handles input from the user should provide a UI to dismiss any unwanted information.

1.3.1.6 In the event of a missing, invalid or partially invalid SPECIFICATION, the system should provide the user with options for exiting the system or fixing the problem e.g., offering defaults for values or offering to direct the user to the authoring interface. If directing the user to the authoring interface, use the Authoring Foundational USAP with parameter SPECIFICATION= SPECIFICATION, APPROPRIATENESS-INFORMATION= APPROPRIATENESS-INFORMATION).

1.3.1.6.1.1.1 Rationale

Even though the portion of the system that creates a new SPECIFICATION provides defaults, a SPECIFICATION might have been corrupted.

A system cannot work without a valid SPECIFICATION.

The user wants to use the system, so if the system can't be used for lack of a valid SPECIFICATION then the user wants to fix it.

1.3.1.6.1.1.2 Implementing this responsibility

The portion of the system that renders information to the user should display information about the options.

The portion of the system that handles input from the user should allow the user to select one of the options.

The portion of the system that validates the SPECIFICATION should generate defaults for unspecified or invalid parameters. These defaults should be consistent with the defaults generated by the portion of the system that creates a new SPECIFICATION.

The portion of the system that validates the SPECIFICATION should act on the option selected by the user.

1.3.1.7 The system must provide a means for displaying and dismissing the content of a SPECIFICATION. (This is so they can decide between multiple SPECIFICATIONs or check that the current SPECIFICATION has correct values.) Optionally, when displaying, the system could offer the user access to the authoring interface to modify parameter values. If directing the user to the authoring interface, use the Authoring Foundational USAP with parameter

SPECIFICATION= SPECIFICATION, APPROPRIATENESS-INFORMATION= APPROPRIATENESS-INFORMATION).

1.3.1.7.1.1.1 Rationale

Users might want to check the parameter values in the SPECIFICATION.
Users might want to modify the parameter values in the SPECIFICATION.

1.3.1.7.1.1.2 Implementing this responsibility

The portion of the system that retrieves the SPECIFICATION must provide the parameters and their values.
The portion of the system that renders information to the user should display the parameters and their values.
The portion of the system that handles input from the user should provide a UI to request the display and dismiss any unwanted information.
Optional connection to the authoring system for users with authoring permission:
The portion of the system that handles input from the user should provide a UI to invoke the authoring system (with parameter SPECIFICATION).
The portion of the system that retrieves the SPECIFICATION should invoke the authoring system (with parameter SPECIFICATION).

1.3.2 Use specified parameters

1.3.2.1 The system must apply the specified parameters as necessary for execution. That is, the items specified must be treated as parameters by the system code (i.e., not hard-coded anywhere) and the values must be taken from the SPECIFICATION. (If logging of execution with specified parameters is desired see Logging Foundational USAP with parameters CONTEXT=Execution.)

1.3.2.1.1.1.1 Rationale

The entire point of this foundational USAP is that users want to control the behavior of the computer in certain ways under certain circumstances that they have previously specified. Therefore, the system must use the specified parameters.

1.3.2.1.1.1.2 Implementing this responsibility

There must be a portion of the system where the parameters that have been specified have some effect.
This portion of the system must treat the parameters as variables rather than as hard-coded values. It must assign the specified parameter values to these variables.

1.3.2.2 The system must provide a UI to accept operator inputs as necessary. (If logging of operator input is desired see Logging Foundational USAP with parameters CONTEXT=Execution.)

1.3.2.2.1.1.1 Rationale

Some of the actions of the SPECIFICATION may require operator input.

1.3.2.2.1.1.2 Implementing this responsibility

The portion of the system that renders information to the user should display the request/opportunity for operator input.

The portion of the system that handles input from the user should provide a UI to provide operator input.

There must be a portion of the system that receives and interprets operator input.

1.4 Logging

Purpose:

- The Logging foundational USAP's purpose is to retain and examine selected information generated during execution.

Justification:

- Some information known only during execution needs to be retained either for debugging or audit-trail purposes.

Glossary:

- To be completed after user testing

Parameters needed by this USAP:

- CONTEXT: The activities that generate the events that are logged.

Foundational USAPs used by this USAP:

(Optionally) Authoring Foundational USAP:

- SPECIFICATION: Logging specification.
- APPROPRIATENESS-INFORMATION: Logging state.
  Shared Assumptions:
  Shared with any other USAP that uses the Logging Specification:
  - The development team has defined the syntax and semantics for the concepts that are included in the Logging Specification.
  - The protocol for saving the Logging Specification is defined and known to the developers of both the authoring system and the systems informed by other USAPs that share the Logging Specification.
  - USAPs sharing these assumptions
    - Authoring Foundational USAP
    - Execution with Authored Parameters Foundational USAP

(Optionally) Execution with Authored Parameters Foundational USAP:

- SPECIFICATION: Logging specification.
- APPROPRIATENESS-INFORMATION: Logging state.
  Shared Assumptions:
  Shared with any other USAP that uses the Logging Specification:
  - The development team has defined the syntax and semantics for the concepts that are included in the Logging Specification.

- The protocol for saving the Logging Specification is defined and known to the developers of both the authoring system and the systems informed by other USAPs that share the Logging Specification.
- USAPs sharing these assumptions
    - Authoring Foundational USAP
    - Execution with Authored Parameters Foundational USAP

USAPs that use this Foundational USAP:
Authorization foundational USAP has two potential contexts that might produce events to be logged

- CONTEXT: Identification.
- CONTEXT: Authentication.

Authoring foundational USAP

- CONTEXT: Authoring.

Execution with Authored Parameters foundational USAP

- CONTEXT: Execution.

1.4.1 **Specify the items to be logged.** This could be done either during development (in which case, this is beyond the scope of this USAP) or during or after deployment (in which case, use the Authoring Foundational USAP with parameter SPECIFICATION=Logging specification, APPROPRIATENESS-INFORMATION=Logging state). Ensure that sufficient parameters are specified so that subsequent analysis is meaningful (e.g., CONTEXT, parameter name, and time stamp). Consider prototyping and testing log information and analysis to ensure sufficiency.

1.4.1.1.1.1.1 Rationale

Software must know the information to be logged.
The values in the repository are going to be examined at a later time and these values must be able to be uniquely identified with sufficient information to be useful.

1.4.1.1.1.1.2 Implementing this responsibility

If done during development:
Implementation is beyond the scope of this USAP.
If done during or after deployment:
Use the Authoring Foundational USAP with parameter SPECIFICATION=Logging specification.

1.4.2 Log items during execution

1.4.2.1 Have a repository in which to store logged items. This repository could be bounded in size, e.g., circular buffer, or unbounded, e.g., disk file.

1.4.2.1.1.1.1 Rationale

The system must have a place to put logged information.

1.4.2.1.1.1.2 Implementing this responsibility

There must be a portion of the system that logs information.
The portion of the system that logs information must know the form of the repository and its location and may be responsible for creating the repository.

1.4.2.2 Enter values into the repository as specified.
(If a Logging specification has been Authored, then use the Execution with Authored Parameters Foundational USAP with parameters SPECIFICATION=Logging specification and APPROPRIATENESS-INFORMATION=Logging state, to access the appropriate specification 1.3.1 and use it to enter the values into the repository 1.3.2.)

1.4.2.2.1.1.1 Rationale

Users need the logged values for debugging or audit trail purposes.
Stored information must persist long enough for analysis to be undertaken.

1.4.2.2.1.1.2 Implementing this responsibility

The portion of the system that logs information enters the particular values into the repository.
Attention should be paid to performance considerations since this code may be executed many time.

1.4.3 Post-processing

1.4.3.1 Retrieve items from the repository. This is typically done some time after the information has been logged, e.g., during the analysis of an anomaly.

1.4.3.1.1.1.1 Rationale

Users need information to analyze past events.
The information they need has been stored in the repository and must be retrieved.

1.4.3.1.1.1.2 Implementing this responsibility

There must be a portion of the system that knows how to get information out of the repository and does so.

1.4.3.2 Support analysis of retrieved items by a log analyst (a special type of user)

1.4.3.2.1.1.1 Rationale

Users need support to analyze past events.

1.4.3.2.1.1.2 Implementing this responsibility

This may be quite complicated (e.g., graphical display, manipulation, mathematical modeling, debugging) and is beyond the scope of this USAP.

2 End-User USAPs

For each End-User USAP. We expect to include

Scenario:

- A story from the end-user's perspective showing the purpose of the end-user USAP.

Usability Benefits:

- Justification for this end-user USAP in terms of usability benefits potentially achieved by implementing this USAP.

For each Foundational USAP used by this USAP (if any), provide:

Parameters furnished to the foundational USAP:

- The parameter values furnished to the foundational USAP used by this USAP.

2.1 Environment Configuration

Scenario:

- An organization wants to supply the same software system to different hardware environments containing different collections of sensors and actuators. A configuration description of the sensors and actuators will allow the system to operate correctly in its environment.

Overview:

- For a software system to be configurable for different environments, actions of the system must be parameterized and the parameter values have to be available at execution time. The values of the parameters must be specified, this configuration description has to be associated with its environment, and the configuration description has to be persistent across sessions.

Glossary:

- To be completed after user testing

Usability Benefits:

- Environment configuration prevents mistakes by tailoring the interface to present only information relevant to the current environment.

Assumptions:

1. There is at least one user who is authorized to author configuration descriptions.
2. The syntax and semantics for the concepts that are included in the configuration description are defined and known to the development team.
3. The protocol for saving the configuration description is defined and known to the development team.
4. Defaults exist for the specifiable parameters.
5. A template exists for authors to use when creating a new configuration description (i.e., the names and definitions of specifiable parameters and their defaults, with optional format).

Foundational USAPs used by this USAP:

Authoring Foundational USAP:

- SPECIFICATION: Configuration description.
- APPROPRIATENESS-INFORMATION: Environment identity

Execution with Authored Parameters Foundational USAP:

- SPECIFICATION: Configuration description.
- APPROPRIATENESS-INFORMATION: Environment identity

2.1.1 Author Configuration Description

Use Authoring Foundational USAP with SPECIFICATION= Configuration description, APPROPRIATENESS-INFORMATION= Environment identity.

Additional responsibilities beyond those inherited from foundational USAPs

None.

Specializations are as follows.

None.

2.1.2 Execute with Authored Configuration Description

Use Execution with Authored Parameters Foundational USAP with SPECIFICATION= Configuration description, APPROPRIATENESS-INFORMATION= Environment identity.

Additional responsibilities beyond those inherited from foundational USAPs

None.

Specializations are as follows.

1.3.1.1Retrieve APPROPRIATENESS-INFORMATION (which will allow determination of appropriate SPECIFICATION) (If APPROPRIATENESS-INFORMATION depends on the user then the user must be authorized. See Authorization USAP with parameters USER=End user and ACTIVITY=Execute the system with parameter values from SPECIFICATION.)

For the Environment Configuration End-User USAP, the APPROPRIATENESS-INFORMATION=environment identity, so there is no need for authorization. In many cases, APPROPRIATENESS-INFORMATION may not need to be retrieved at all because the location of the configuration description is built-in (e.g., config.dat at a known location). If so, this responsibility is not applicable.

1.3.1.4 The system must check that the retrieved SPECIFICATION is valid for use.

For the Environment Configuration End-User USAP, the validity depends on the consistency between the Configuration Description and the physical reality of execution environment. Thus, consistency involves checking hardware. For example, sensors may need to be polled to verify that they are currently present and working.

2.2 User Profile

Scenario:

- A user wishes to have the capabilities of the system personalized to reflect his or her preferences or role. The capabilities that can be personalized may include language, access to system functionality, display characteristics, account information or any preference that might vary among users or roles.

Overview:

- For a user profile to work, configurable actions of the system must be parameterized and the parameter values have to be available at execution time. The values of the parameters must be specified, this specification (the "user profile") has to be associated with the user and persist across sessions.

Glossary:

- To be completed after user testing

Usability Benefits:

- User profile accelerates error-free portion of routine performance by providing information in a familiar and individually-tailored form, providing user-defined hot-keys and allowing common operations to be easily accessible.
- User profile prevents mistakes by simplifying the interface to that which is familiar and necessary. This prevents infrequent users from making the types of mistakes caused by too many options.
- User profile accommodates mistakes by enabling the user to disallow certain options (e.g., disabling the tap-to-click option on a track pad).
- User profile increases user confidence and comfort by providing an individualized interface.

Assumptions:

1. There is at least one user who is authorized to author user profiles.
2. The syntax and semantics for the concepts that are included in the User Profile are defined and known to the development team..
3. The protocol for saving the User Profile is defined and known to the development team.
4. Defaults exist for the specifiable parameters.
5. A template exists for authors to use when creating a new user profile (i.e., the names and definitions of specifiable parameters and their defaults, with optional format).

Foundational USAPs used by this USAP:

Authoring Foundational USAP:

- SPECIFICATION: User profile
- APPROPRIATENESS-INFORMATION: User identity.

Execution with Authored Parameters Foundational USAP:

- SPECIFICATION: User profile
- APPROPRIATENESS-INFORMATION: User identity

2.2.1 Author User Profile

Use Authoring Foundational USAP with SPECIFICATION=User Profile, APPROPRIATENESS-INFORMATION=User identity.

Additional responsibilities beyond those inherited from foundational USAPs

None.

Specializations are as follows.

None.

2.2.2 Execute with Authored User Profile

Use Execution with Authored Parameters Foundational USAP with SPECIFICATION=User Profile and APPROPRIATENESS-INFORMATION=User identity.

Additional responsibilities beyond those inherited from foundational USAPs

None.

Specializations are as follows.

1.3.1.1Retrieve APPROPRIATENESS-INFORMATION (which will allow determination of appropriate SPECIFICATION) (If APPROPRIATENESS-INFORMATION depends on the user then the user must be authorized. See Authorization USAP with parameters USER=End user and ACTIVITY=Execute the system with parameter values from SPECIFICATION.)

For the User Profile End-User USAP, the APPROPRIATENESS-INFORMATION=user identity, so use the AUTHORIZATION foundational USAP with USER=End user and ACTIVITY=Execute the system with parameter values from user profile.

2.3 Alarms and Events

Scenario:

The user needs feedback from the system when an error occurred or a specific condition is met. The user can be the operator of the system or a superior system. The feedback can be needed for safety reasons, diagnostic, problem solving or information purposes.

Overview:

Alarm, Events and Messages Management System

EEMUA describes how Alarm & Events are important in the control of plant and machinery. Alarm and Event systems form an essential part of the operator interfaces in large modern industrial systems. They provide vital support to the operators managing these complex systems by warning them of situations that need their attention.

Alarms are signals which are annunciated to the operator typically by an audible sound, some sort of visual indication, usually flashing, and by the presentation of a message or some other identifier. An alarm will indicate a problem requiring operator attention, and is generally initiated by a process measurement passing a defined alarm setting as it approaches an undesirable or potentially unsafe value. Alarm systems help the operator;

- to maintain the plant within safe operating envelope;
- to recognize and act to avoid hazardous situations;
- to identify deviations from desired operating conditions that could lead to financial loss;
- to better understand complex process conditions. Alarms should be an important diagnostic tool, and are one of several sources that an operator uses during an upset.

The terms alarm and event are often used interchangeably and their meanings are not distinct. An alarm is an abnormal condition that requires special attention. An event may or may not be associated with a condition. For example, the transitions into the level alarm condition and the return to normal are events which are associated with conditions. However, operator actions, system configuration changes, and system errors are examples of events which are not related to specific conditions.

Alarm processing and handling
The Norwegian Petroleum Directorate(NPD) has identified alarm processing and handling concepts described in Figure 1 and used in this document;

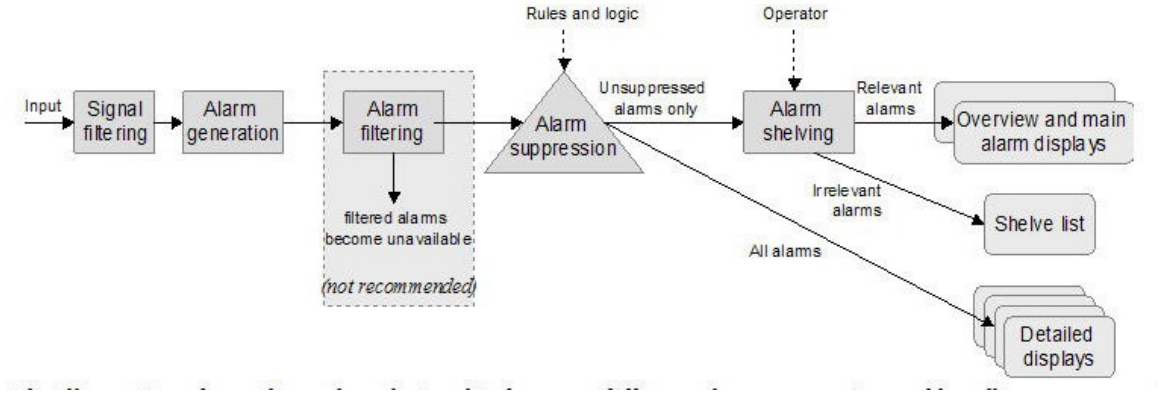


Figure 1. NPD's definition of alarm processing.

- Alarm generation means generating an alarm according to some defined rules.
- Alarm filtering means preventing an alarm signal so that it is not available for the operator in any part of the system.
- Alarm suppression means preventing an alarm from being presented in main alarm displays, e.g. overview displays, but the alarm is still available in the system at a more detailed level.
- Alarm shelving is a facility for manually removing an alarm from the main list and placing it on a shelve list, temporarily preventing the alarm from re-occurring on the main list until it is removed form the shelf. Shelving will normally be controlled by the operator, and is intended as a "last resort" for handling irrelevant nuisance alarms that have not been caught by signal filtering or alarm suppression mechanisms.

Figure 2 shows the actions which cause transitions between the states that a displayed alarm may have according to EEMUA. The terminology is defined in Table 1.

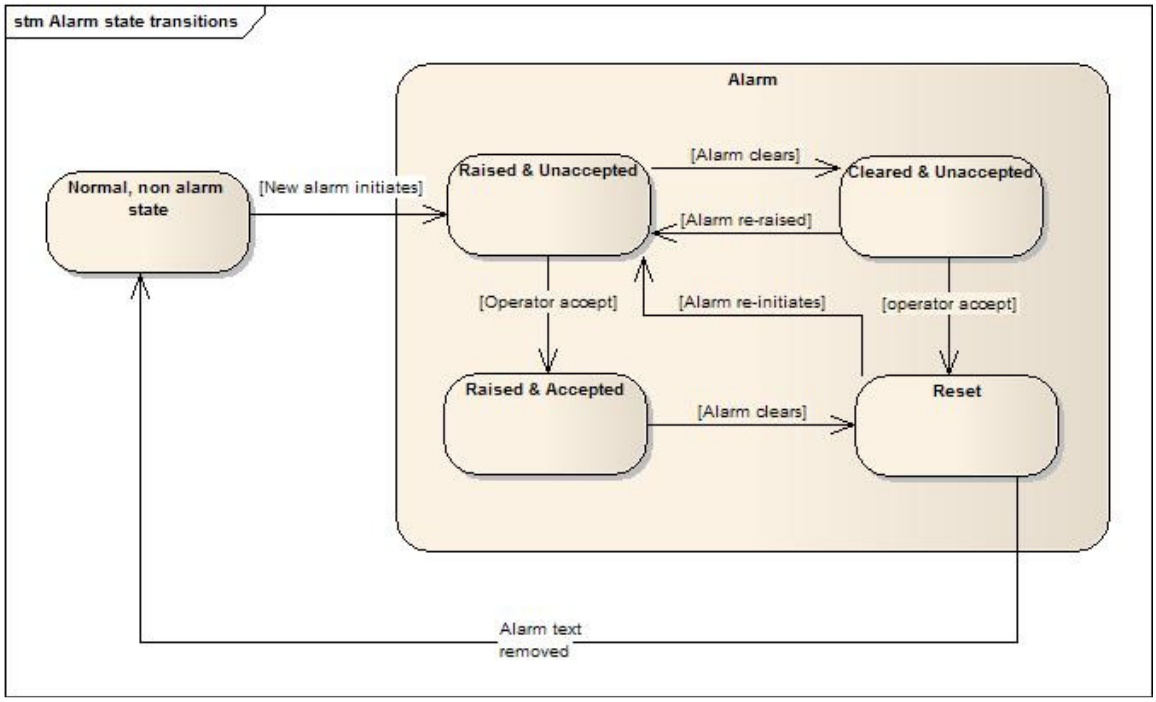


Figure 2. EEMUA's defintions of alarm state transitions.

Other definitions

- Alarm prioritization is a categorization of alarms based on the importance of each alarm for the operator tasks.
- Overview displays are designed to help operators get an overview of the state of the process. Overview displays include: Main alarm lists, tiles or enunciator alarm displays, as well as large screen displays showing key information.
- Selective lists show only a selection of the available alarm information, based on selection and sorting criteria specified by operators. For more definitions see Table 1.

References

[1] EEMUA 191: Alarm Systems. A Guide to Design, Management and Procurement. 1999, ISBN 0 8593 1076 0 (http://www.eemua.co.uk).
[2] Norwegian Petroleum Directorate YA-711: Principles for alarm system design, 2001 (http://www.ptil.no/regelverk/R2002/ALARM_SYSTEM_DESIGN_E.HTM).
[3] MSDN library, Windows Vista User Experience
[4] M.Hollender, Beuthel, C.: Intelligent alarming, Effective alarm management improves safety, fault diagnosis and quality control, ABB Review 1, 2007.

Glossary:

`The following definitions are used in this document.`
`<insert Pia's table)`

Usability Benefits:

- Reduces impact of slips by informing the user about the deviation from normal procedure.
- Supports problem solving by helping user understand the problem and contributes to the solution of the problem.
- Facilitate learning by helping the user to understand and learn the consequences of actions
- Action confirmation feedback prevents the user from going any further with a mistake. The feedback acts as a gatekeeper, and need an acknowledgement to go ahead with the procedure.
- Feedback helps user tolerate system error by informing the user that there is a problem and possible cause of the problem.
- Being able to solve a problem with help of a system's feedback will increase user confidence by supporting the user performing his/her work. It will also build trust for the system as it always presents accurate and helpful feedback.

Assumptions:

1 There is at least one user who is authorized to author rules for alarms, events and display.
2 The syntax and semantics for the concepts that are included in the Alarms&Events specification are defined and known to the development team. These concepts may include values, logic, properties, etc. to control the behavior of alarms and events. These concepts must support the customers current alarm philosophy and the relevant standards that apply for the systems customers. See AlarmLanguage_BassJohnGolden.doc for an example inspired by the Norwegian Petroleum Directorate YA-711: Principles for alarm system design, 2001 (http://www.ptil.no/regelverk/R2002/ALARM_SYSTEM_DESIGN_E.HTM).
3 The development team has defined presentation conventions that are salient to the user, e.g, fonts, icons, colors. These conventions must differentiate alarms, events, and messages from each other.
4 The protocol for saving an Alarms&Events specification is defined and known to the development team.
5 There is no more than one Alarms&Events specification per context of use. That is, all of the rules are bundled into one specification that the system loads when it is executed in a particular context. So, for example, there may be a single specification for normal operation of the system and a different single specification for diagnosis procedures.
6 The development team has decided on an appropriate protocol for concurrent users. That is, can each user be completely autonomous (e.g., being able to dismiss or suppress alarms), is one user the "master" and all the rest can only observe, etc.
7 The development team has decided which items will be logged during execution. For example, these items might include: state transitions of an alarm, the event of failure of the alarm generation routine, the alarm generation rate,

Foundational USAPs used by this USAP:

Authoring Foundational USAP:

- SPECIFICATION: Rules for Alarms, Events and Displays

- APPROPRIATENESS-INFORMATION: Context of use.

Execution with Authored Parameters Foundational USAP:

- SPECIFICATION: Rules for Alarms, Events and Displays
- APPROPRIATENESS-INFORMATION: Context of use.

2.3.1 Author Rules for Alarms, Events and Displays

Use Authoring Foundational USAP with SPECIFICATION=Rules for Alarms, Events and Displays, APPROPRIATENESS-INFORMATION= Context of use. By "Context of use" we mean things like "in operation" "in maintenance" "in diagnosis".

Additional responsibilities beyond those inherited from foundational USAPs

None.

Specializations are as follows.

1.2.1.1 The system must provide a way for an authorized author to create a SPECIFICATION. (See Authorization Foundational USAP with parameters USER=Author and ACTIVITY=Author the SPECIFICATION.)
For the Alarms and Events End-User USAP, there is an assumption that a language has been defined that describes the parameters and their interactions (Assumption 1). The specification being created must conform to this language.

1.2.2 Save a SPECIFICATION. The system must provide a means for the SPECIFICATION to be saved and/or exported (e.g., by autosave or by author request). If other systems are going to use the SPECIFICATION, then use a format that can be used by the other systems. (If logging of authoring results is desired see Logging Foundational USAP with parameter CONTEXT=Authoring.)
For the Alarms & Events End-User USAP logging authoring results may be needed for regulatory purposes. If this is the case, then use the Logging Foundational USAP with Context=Authoring.

1.2.3.2 Display the current parameter values (including identifier and circumstances).
For the Alarms & Events End-User USAP the system must not only display the current parameter settings, but also display some of the implications of some of the current parameter settings. For example, the system must display the current priority distribution of the alarms during authoring of the setting of the alarm priority. E.g. "5% of the configured alarms are high priority". [Section 2.5. #27, p. 15 Norwegian Petroleum Directorate YA-711: Principles for alarm system design, 2001 (http://www.ptil.no/regelverk/R2002/ALARM_SYSTEM_DESIGN_E.HTM).

1.2.3.3 The system must provide a ways for an authorized author to change the parameter values…
For the Alarms & Events End-User USAP the authorization information to modify a rule is a property of the rule. Therefore use the Authorization

Foundational USAP with parameters USER=Author and ACTIVITY=Modify a particular rule.

1.1.3.1 The system must permit or prohibit specific ACTIVITIES dependent on who the USER is and on the set of ACTIVITIES they are attempting to perform, i.e., the system must know (built in, stored, provided by the OS, etc.) the specific permissions for each USER and then enforce these permissions
For the Alarms & Events End-User USAP the permission information to modify a rule is a property of a rule and the system must know how to retrieve the permission information from that rule.

2.3.2 Execute with Authored Rules for Alarms, Events and Displays

Additional responsibilities beyond those inherited from foundational USAPs

2.3.2.1 The system must permit multiple users to operate simultaneously in accordance with the protocol defined in the assumptions, e.g., each user could have the ability to have their own display filter settings.

2.3.2.1.1.1.1 Rationale

Some large systems may require more than one operator to function safely.
Each operator may want to sort the alarm list display according to different criteria depending on his/hers current task or preferences.

2.3.2.1.1.1.2 Implementing this responsibility

The portion of the system that stores system data must be shareable among multiple users.
The portion of the system that manages system data must synchronize among multiple users to avoid simultaneous update of system data.
The portion of the system that manages system data must implement a protocol that determines what the system must do in the event that two users simultaneously issue conflicting commands. Consider informing the users in the event of a conflict as a portion of the protocol.
The portion of the system that manages user-specific data must be thread safe (e.g., re-entrant)
The portion of the system that interacts with the users must be thread safe.

2.3.2.2 The system must have the ability to translate the names/ids of externally generated signals, e.g., from a sensor, into the concepts that are included in the Alarms&Events specification.

2.3.2.2.1.1.1 Rationale

The environment contains sensors that generate and actuators that respond to analog or digital signals in their own form.
The alarm and event portion of the system can only operate with logical concepts.

2.3.2.2.1.1.2 Implementing this responsibility

There should be a portion of the system (e.g., an intermediary) that sits between those portions of the system that interact directly with sensors and actuators (e.g., the device drivers) and the portion of the system that implements the alarm and event logic.
This intermediary should translate between the signals by the sensors and actuators and the logical concepts required by alarm and event rules.

2.3.2.3 The system must have the ability to broadcast a generated event so that an external system can use it. E.g. an external long-time storage system.

2.3.2.3.1.1.1 Rationale

Some events require informing external people or systems. For example, an explosion may need to call emergency responders.

2.3.2.3.1.1.2 Implementing this responsibility

The portion of the system that executes the alarm and event rules must have the capability to broadcast to appropriate external systems.

2.3.2.4 The system must have the ability to present alarm state transitions in the alarm displays within the time restrictions valid for this system.

2.3.2.4.1.1.1 Rationale

Users have limits as to how fast they can operate, i.e., perceive information, comprehend information, make decisions, and perform motor actions. This imposes a lower bound on how long information has to be displayed (visual or auditory).
Because the alarm system may supervise hazardous environments, safety regulations may require specific response times. This imposes an upper bound on the number of human actions that can be required to respond to alarms.
Any claims made for the operator action in response to alarms should be based upon sound human performance data and principles.

2.3.2.4.1.1.2 Implementing this responsibility

The portion of the system that displays information to the user should ensure that information is displayed long enough for a person to see or hear it. This requires that this portion of the system maintain timing information of how long information has been displayed and ensure that it is longer than minimal human perceptual limits.
The portion of the system that does the scheduling should schedule alarms as high-priority activities.
The portion of the system that interacts with the user during emergency situations must be designed to meet the response time requirements. This is the responsibility of the UI designers and does not impose additional architectural requirements.

2.3.2.5 The system must have sufficient persistent storage for alarms, rules

and data to be saved. It may be acceptable to limit the number of events to be saved.

2.3.2.5.1.1.1 Rationale

Regulations might require long-term storage of alarms, rules and data (e.g. in the food and drug business).
An organization may have publication, notification, history, fault diagnostic needs that require persistent data.
The system may be stopped and started again and the rules and data must not get lost when this happens.
There might be a lot of data.
Storage media has cost (hardware, time to read and write, network bandwidth).

2.3.2.5.1.1.2 Implementing this responsibility

The portion of the system that manages persistent data must ensure that the most recent and the most important data is not lost. This could be done by having large persistent data stores; it could be done by overwriting older data with newer data.
The portion of the system that manages persistent data must have a protocol to determine which data gets overwritten when persistent storage is almost full.

Specializations are as follows.

1.3.1.1 Retrieve APPROPRIATENESS-INFORMATION (which will allow determination of appropriate SPECIFICATION) (If APPROPRIATENESS-INFORMATION depends on the user then the user must be authorized. See Authorization USAP with parameters USER=End user and ACTIVITY=Execute the system with parameter values from SPECIFICATION.)
For the Alarms & Events End-User USAP the APPROPRIATENESS-INFORMATION may include not only the Context of use, but in the case of concurrent users also who is using a display. Therefore also use the Authorization Foundational USAP with parameters USER=End-user and ACTIVITY=Execute with Rules for Alarms, Events and Displays.

1.3.2.1 The system must apply the specified parameters as necessary for execution. That is, the items specified must be treated as parameters by the system code (i.e., not hard-coded anywhere) and the values must be taken from the SPECIFICATION. (If logging of execution with specified parameters is desired see Logging Foundational USAP with parameters CONTEXT=Execution.)
For the Alarms & Events End-User USAP, the information to be logged must be in accordance with the assumptions about logged information. Therefore use the Logging Foundational USAP with parameters CONTEXT=Execution.

Other things to consider when designing alarms, events and displays

- Provide the ability to support native languages.
- Provide context-sensitive help for instances of alarms, events and messages guided by their unique specification identity.
- Present multiple views of the alarm displays. For example, one view could include N

number of raised alarms, another could include all raised alarms since the timestamp of the oldest raised and non-cleared alarm.

- Give feedback on user actions within 150 ms. Feedback must be appropriate to the manner in which the command was issued. For example, if the user pressed a button, changing the color of the button would indicate the user feedback.