

APSP¹

Abraham James², Christopher Rebstock³, Kareemullah Quadir⁴,
Roshan Rammohan⁵, Vinod Doshi⁶, Yaroslav Halchenko⁷
<http://apsp.onerussian.com>

February 18, 2002

¹Automated Personal Software Process

²ajames@cs.unm.edu

³chrisr@cs.unm.edu

⁴quadir@cs.unm.edu

⁵roshan@cs.unm.edu

⁶vdoshi@cs.unm.edu

⁷yoh@cs.unm.edu

Chapter 1

Business Case

Already presented - omitted for now.

Chapter 2

Requirements

2.1 Introduction

The Personal Software Process [2] is a software development process that allows the individual to apply industrial level discipline to his or her practice.

Our objective is to develop a APSP (Automated Personal Software Process) tool, which would be used together with CVS to automate the PSP process to a large extent. Tasks like monitoring and reporting (log, statistics, progress) for each individual software developer would require no manual intervention.

To assess personal achievements and progress it is useful to have a tool (examples: [1]) which would show trends of stages where developer spends most of the time (developing, debugging, learning new technology and tools etc), provide statistics on how code-development process going on (how many lines were added/removed/modified).

2.2 Use Cases

2.2.1 Use Case “SystemInit”: Startup and Initialization of System

Goal in Context

System is initialized, the path(s) to projects and refresh time (time interval between checking for changes in project files) are set.

Preconditions

1. User has system properly installed.
2. Configuration file was properly created.

Success End Conditions

System starts successfully and begins authentication.

Failed End Conditions

System did not start.

Primary Actor

User

Trigger

User executes system. System is executed from automated login script.

Description (Main Success Scenario)

1. User executes system.
2. System verifies correctness of configuration file
3. System begins authentication. (use case “user authentication”)

Extensions

- 1a The system is not installed properly and gives error upon execution.
- 1b The configuration file is not found. The system informs the user that the configuration file was not found
- 1c A path is not set. The system informs the user that a path must be set.
- 1d The refresh time is not set. The system informs the user that the default value will be used

2.2.2 Use Case “RegisterAccount”: Register an account

Goal in Context

Allow the user to register an account with the system.

Preconditions

Success End Conditions

Account for a new user created.

Failed End Conditions

Account wasn't created.

Primary Actor

User

Trigger

User issues request to register new account

Description (Main Success Scenario)

1. User issues request to register new account
2. System asks for personal information: login, password and email address.
3. User enters the required information.
4. User submits request to register an account.
5. System validates the user and creates an account.

Extensions

- 4a A valid email wasn't entered. The user is prompted to re-enter a valid email. Email validation also possible.
- 4b User name already exists in the system. The user is prompted to enter another one.
- 4c Entered password didn't validate. The user is prompted to reenter
- 5a Can be done automatically through email validation.

2.2.3 Use Case “UserAuth”: User authentication

Goal in Context

Authenticate user so as to protect the user’s data.

Preconditions

User has a valid account with system.

Success End Conditions

User is authenticated and system begin logging in data.

Failed End Conditions

User authentication fails.

Primary Actor

User

Trigger

The user initiates request for authentication.

Description (Main Success Scenario)

1. The system prompts the user for username and password.
2. The system validates the entered information.
3. The user is given access to the system and the data.

Extensions

- 1a The username and/or password is incorrect, the user is prompted to re-enter the information.

2.2.4 Use Case “LogOut”: User Logs Out

Goal in Context

Shutdown the system and stop all logging.

Preconditions

User is currently logged in

Success End Conditions

Logging is stopped and user’s status is set to logged out.

Failed End Conditions

- System crashes and begins logging erroneous data.
- System crashes and is no longer able to collect data.

Primary Actor

User

Trigger

User issues command shutdown.

Description (Main Success Scenario)

1. User initiates shutdown command.
2. System stops all logging.

2.2.5 Use Case “DeleteAccount”: Delete user account

Goal in Context

Allow administrator to be delete user account.

Preconditions

User account exists.

Success End Conditions

Terminated the account. System deletes the account and denies access.

Failed End Conditions

Account wasn't deleted.

Primary Actor

Administrator.

Trigger

Administrator issues command to delete user account.

Description (Main Success Scenario)

1. Administrator logs in to the system.
2. Administrator deletes specified user account.

Extensions

- 1a No valid user account for specified login name.

2.2.6 Use Case “ChangePassword”: Change of User’s Password

Goal in Context

Allow user to changes his password.

Preconditions

1. User has valid account.
2. User knows his current password.
3. User is currently logged in.

Success End Conditions

User changed his password to a new one, which he specified.

Failed End Conditions

Password wasn’t changed.

Primary Actor

User

Trigger

User issues command change password.

Description (Main Success Scenario)

1. User issues command change password.
2. System asks old password.
3. User enters old password.
4. System asks new password.
5. User enters new password.
6. System asks user to re-enter new password.
7. User re-enters new password.
8. System changes user’s password to the new one.

Extensions

- 3a Typed in old password is wrong, system informs the user and goes back to the step 2.
- 3b Typed in new password differs from the typed in password in verification field. System informs the user and goes back to the step 2.

Variations

- User may cancel transaction at any time before step 4, so passwords wouldn't change

2.2.7 Use Case “ForgotPassword”: User forgot his password

Goal in Context

Give possibility to provide the user with a new password for his account, trying to be secure enough.

Preconditions

1. User has a valid account with system.
2. User doesn't know his current password and hence cannot login to the system.
3. User provided a valid email address when account was created.

Success End Conditions

User gets new temporary valid password to his account.

Failed End Conditions

User did not obtain a new password.

Primary Actor

User

Trigger

User initiates request for a new password.

Description (Main Success Scenario)

1. User provides the system with login name and valid email address for that login name.
2. System validates entered email address.
3. System changes password for that account to a new, randomly generated one.
4. System sends the new password to the specified email address

Extensions

- 2a If entered email address doesn't correspond to his login name - system informs the user and returns to the step 1.
- 4a E-mail address expired and no longer valid, so user's password can't be changed this way - failed end condition.

Variations

- 5a System asks user to change his new password to another one as soon as logs into the system. (Previous Use Case: Change of the Password.)

2.2.8 Use Case “FillDRC” : User completes the Design Review Checklist.

Goal in Context

To provide the user a way to fill out an effective design review checklist.

Preconditions

User is logged in.

Success End Conditions

User submits the Design Review Checklist and system is updated successfully.

Failed End Conditions

No active projects for the user - no system update.

Primary Actor

User

Trigger

User issues the command open DRC.

Description (Main Success Scenario)

1. User issues the command to open new DRC.
2. User specifies a project for which project this DRC is.
3. System opens a new Design Review Checklist.
4. User completes the Design Review Checklist and submits it.
5. System is updated successfully.

2.2.9 Use Case “SwitchPhase”: User switches between project phases.

Goal in Context

User wishes to switch from one project phase to another.

Preconditions

User is currently logged in.

Success End Conditions

1. The change of phase is indicated to the user.
2. The system starts recording data for the new phase.

Failed End Conditions

The system does not switch to the new phase.

Primary Actor

User

Trigger

User gives a command to change to another phase.

Description (Main Success Scenario)

1. User gives a command to change to another phase.
2. System changes the phase.
3. System indicates the change of phase to the user and starts recording data for the new phase.

Extensions

None.

2.2.10 Use Case “AddEditPhaseData”: Add/Edit project phase data.

Goal in Context

User wishes to add/edit existing information of a particular phase.

Preconditions

User is currently logged in.

Success End Conditions

The user added/edited project phase data.

Failed End Conditions

The user could not add/edit the data.

Primary Actor

User

Trigger

User gives a command to add/edit data of a project phase.

Description (Main Success Scenario)

1. User gives a command to add/edit data of a project phase.
2. System opens the project phase.
3. User adds/edits data for that phase and submits it.
4. System updates the data for that phase successfully.

2.2.11 Use Case “SubmitDefects”: User submits Defect recording Log (DRL)

Comment

System will provide basic statistics of number of defects detected/solved during development. It will not be provide full capability of defect tracking system.

Goal in Context

To hold data on each defect as the user finds and corrects it.

Preconditions

User is currently logged in.

Success End Conditions

User submits the defects and system is updated successfully.

Failed End Conditions

Primary Actor

User

Trigger

User issues the command open DRL.

Description (Main Success Scenario)

1. User issues the command open DRL.
2. System opens a new DRL.
3. User completes the log and submits it.
4. System is updated successfully.

Variations

- 1a User tries to submit an incomplete log and system prompts an error message asking the user to complete the log.

2.2.12 Use Case “FillPPS” : User completes the Project Plan Summary (PPS) form.

Goal in Context

To hold the estimated and actual project data in a convenient and readily retrievable form.

Preconditions

1. User is logged in.
2. User has already submitted the rest of the forms except the PIP form regarding the project.

Success End Conditions

User successfully submits the PPS form and the System is updated.

Failed End Conditions

User submits the form, duly filled, but the system update fails.

Primary Actor

User

Trigger

User submits all the forms and information and gives a command 'open PPS form'.

Description (Main Success Scenario)

1. User issues command open PPS form.
2. System opens the partially filled PPS form.
3. User completes the form and submits it.
4. System is updated.

Extensions

- 2a User tries to submit form with invalid data. System prompts an error message asking the user to reenter information.
- 2b User tries to submit an incomplete form. System prompts an error message asking the user to complete the form.

2.2.13 Use Case “FillPIP” : User submits the Process Improvement Proposal (PIP) form.

Goal in Context

To record process problems and improvement ideas by filling out the PIP form.

Preconditions

1. User is logged in.
2. User has completed the PPS form.

Success End Conditions

User submits the PIP form and system is updated successfully.

Failed End Conditions

User submits the PIP form but system is not updated.

Primary Actor

User

Trigger

User issues command to open PIP form.

Description (Main Success Scenario)

1. User gives command open PIP form.
2. System opens a new PIP form.
3. User enters all the information and submits the form.
4. System is updated successfully.

Extensions

- 2a User tries to submit an incomplete form. System prompts an error message asking the user to complete the form.

2.2.14 Use Case “ViewStat”: User views Graphs.

Goal in Context

User wishes to view the graphs on his progress.

Preconditions

1. User is currently logged in.
2. The system has collected statistical data in projects on his previous projects.

Success End Conditions

User views the graph.

Primary Actor

User.

Trigger

User issues a command to view graphs.

Description (Main Success Scenario)

1. User issues a command to view graphs.
2. System displays list of graphs.
3. User chooses the graph he wishes to view.
4. System displays the previous projects and prompts the user to enter the project range he wishes to view the graph for.
5. User enters the project range.
6. System displays the graph.
7. User goes back to step 1 or exits.

Extensions

None.

Variations

- 5a User need not specify project range - so graph for all previous projects will be shown.

2.3 Domain Model

2.3.1 Classes and Associations

All classes and relationships presented on Fig. 2.1

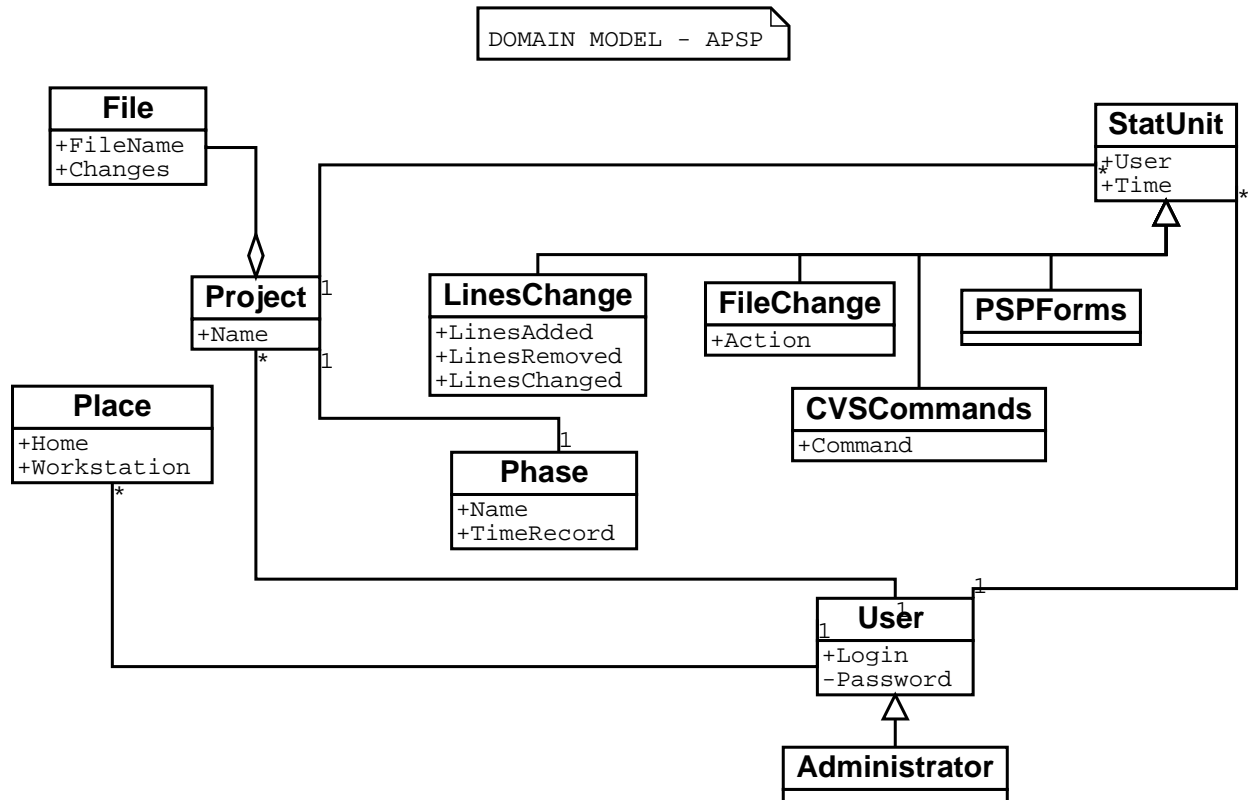


Figure 2.1: Domain Model of APSP

User The User Class contains information specific to users.

Attributes

- Login
- Password

Every instance of User is associated with Place, Project and StatUnit.

Administrator The Administrator Class inherits from the User Class. This class exists to make a distinction and to give special privileges to instances of the Administrator.

Project The Project class contains information about a project that a user is associated with. It is essentially a directory, an aggregation of Files. It is also associated with Phase.

Files The File class contains the filename and the record of changes associated with it. It is associated with StatUnit.

Place The Place class is associated with a User. It holds information about the time that User spends working on the project at probably different locations. For Eg. Home, Different workstations.

StatUnit StatUnit class contains units of statistical information. It is associated with a User and has a record of the time that this data was recorded.

LinesChange LinesChange class inherits from StatUnit. It is designed to hold information specific to Files.

Attributes

- LinesAdded
- LinesRemoved
- LinesChanged

FileChange FileChange class inherits from StatUnit. It is designed to hold information about Files that were added or removed from a particular project.

PSPForms PSPForm also inherits from StatUnit. It holds information collected from the PSP forms that the user fills up.

Phase The Phase class holds information about the time user spends in different phases of software development. Its attributes include Name and TimeRecord.

2.3.2 Queries

ShowPPS

Description The user requests to view the Project Plan Summary for a specific project. The system checks if the specified project exists. If the project exists, the system displays the PPS (Project Plan Summary) form for that project. If the project is current one, then the system displays the partially filled PPS form. If the project does not exist, then the system prompts an error message informing the user that the specified project does not exist.

Inputs Project ID.

Results

1. Project Plan Summary form displayed.
2. Error message that specified project does not exist.

ShowTRL

Description The user requests to view the Time Recording Log for a specific project. The system checks if the specified project exists. If the project exists, the system displays the TRL (Time Recording Log) for that project. If the project does not exist, the system prompts an error message informing the user that the specified project does not exist.

Inputs Project ID.

Results

1. Time Recording Log displayed.
2. Error message that specified project does not exist.

ShowDRL

Description The user requests to view the Defect Recording Log for a specific project. The system checks if the specified project exists. If the project exists, the system displays the DRL (Defect Recording Log) for that project. If the project does not exist, the system prompts an error message informing the user that the specified project does not exist.

Inputs Project ID.

Results

1. Defect Recording Log displayed.
2. Error message that specified project does not exist.

ShowDRC

Description The user requests to view the Design Review Checklist. The system checks if the specified project exists. If the project exists, the system displays the DRC (Design Review Checklist) for that project. If the project does not exist, the system prompts an error message informing the user that the specified project does not exist.

Inputs Project ID.

Results

1. Design Review Checklist displayed.
2. Error message that specified project does not exist.

ShowSET

Description The user requests to view the Size Estimating Template. The system checks if the specified project exists. If the project exists, the system displays the SET (Size Estimating Template) for that project. If the project does not exist, the system prompts an error message informing the user that the specified project does not exist.

Inputs Project ID.

Results

1. Size Estimating Template displayed.
2. Error message that specified project does not exist.

ShowGraph

Description The user requests to view the graph. The user enters the type of graph he wishes to view. The system prompts the user to enter the project range he wishes to view the graph for. The user enters the project range. The system displays the graph. If the user does not enter the type of graph, the system prompts error message asking the user to enter the graph type. If the specified graph does not exist, the system prompts error message informing the user that the specified graph does not exist. If the user does not specify the project range, all the previous projects will be considered. If the project IDs are invalid, the system prompts error message that specified project does not exist.

Inputs Graph type. Project ID range.

Results

1. Specified Graph displayed.
2. Error message that specified graph does not exist.
3. Error message that specified project does not exist.

2.3.3 Actions

Use Case “SystemInit”

User executes system. System responds by beginning the loading process, duration is variable dependant on size and speed of users system.

Use Case “RegisterAccount”

User issues request to register new account. System responds by querying for personal information: login, password and email address. Duration of Action is dependant on how long it takes the user to enter the required information.

Use Case “UserAuth”

System issues a request for username and password. Duration will be short, as the system need only verify the user name and password. The system will then grant or deny access based on the user’s information

Use Case “LogOut”

User initiates command for system shutdown. System responds by doing one final check of the user’s files, then stops all logging and shuts itself down.

Use Case “DeleteAccount”

Privileged user initiates a delete account request. System responds by removing the account from the system entirely. Duration will be dependant on the size of the account being removed.

Use Case “ChangePassword”

User initiates a command to change his password. System responds by prompting him to enter his current password and enter a new password twice. After verifying that both passwords entered were the same the user’s password is changed. Duration should be short, since the system only needs to verify that both fields for the new password are the same.

Use Case “ForgotPassword”

User initiates request for a new password. The user does not need to be logged in to request this password for his account. The system will then email the user a new randomly generated password.

Use Case “FillDRC”

User initiates command to open a new DRC file. System responds by giving the user an empty file to complete. Once all information has been filled in properly the user may submit it to the system for storage.

Use Case “SwitchPhase”

User initiates command to switch phase. System then switches phase and notifies user.

Use Case “AddEditPhaseData”

User requests to manually edit or add a phase. The system responds by opening the project phase. The user completes entry of information and submits it to the system.

Use Case “SubmitDefects”

User enters command to open new DRL. Once the system gives the user the DRL, the user can complete log and submit it.

Use Case “FillPPS”

User initiates request for a PPS form. System gives user a partially filled in form, which the user can fill and submit. Once submitted the system will update its information based on the PPS form. Duration is entirely dependant on how long it takes the user to enter the information.

Use Case “FillPIP”

User initiates request for a PIP form. System gives user the form, which the user can fill and submit. Once submitted the system will update its information based on the PIP form. Duration is entirely dependant on how long it takes the user to enter the information.

Use Case “ViewStat”

User enters command to view graphs. System responds by prompting user with a list of graphs. User selects a graph which is then displayed.

2.3.4 Events

Register an account

When the user issues request to register an account, the system asks for the login,password and email address of the user. Using this information the system creates/registers a new account for the user.

User authentication

User uses his user name and password to login in the system. The system uses this information to log the user in the system.

User Logs Out

After the user issues a command to log out, the system logs the user out of the system.

Delete user account

The Administrator deletes a particular account from the system. The account no longer exists in the system.

Change of User’s Password

User changes his password. The old password and the new password are given as an input to the system. The password of that user is changed.

User completes the Design Review Checklist

The user fills out a design review checklist. The system updates this data successfully.

User switches between project phases.

User gives a command to switch from one phase to another. The system switches to the new phase and starts recording data for the new phase.

Add/Edit project phase data.

User gives a command to add/edit the data related to a phase. The system saves the data added/edited by the user.

User submits Defect recording Log (DRL)

User submits the defects and system is updated successfully.

User completes the Project Plan Summary (PPS) form.

User submits the data for the PPS and the system saves the data successfully.

User submits the Process Improvement Proposal (PIP) form.

User submits the data for the PIP and the system saves the data successfully.

2.3.5 State Transitions

It was pretty hard to find objects in our system which would have interesting states transitions. So **User** class can be in states as LoggedOff, Debugging etc, but here we have case with all-to-all kind of transitions, that is why it is out of interest.

StatUnit can be in states Occurred, Monitored and Stored and transitions between them are sequential, so it is also out of interest.

So we decided to work out general transitions diagram for whole system (Fig. 2.2).

Most of the time, since user logs in, system resides in idle state. Periodically (on timer) it checks dependent files on presence of changes in them, if such exists - system stores that changes in inner database. Other request to log some activities (cvs commands, user changed phase etc) can cause system to store them too.

All user cases are joint in this phase transition diagram in "UI" state - when system needs to gather some information from the user, and depending on that interaction it can disconnect, store some data entered by user (PSP forms, phase change etc) in the system or just simply return in Idling state.

Current state diagram has some limitations, because it shows how system would look in case of sequencing programming - without multithreading, which is very easy to utilize, so to make system more tolerant to the user activity.

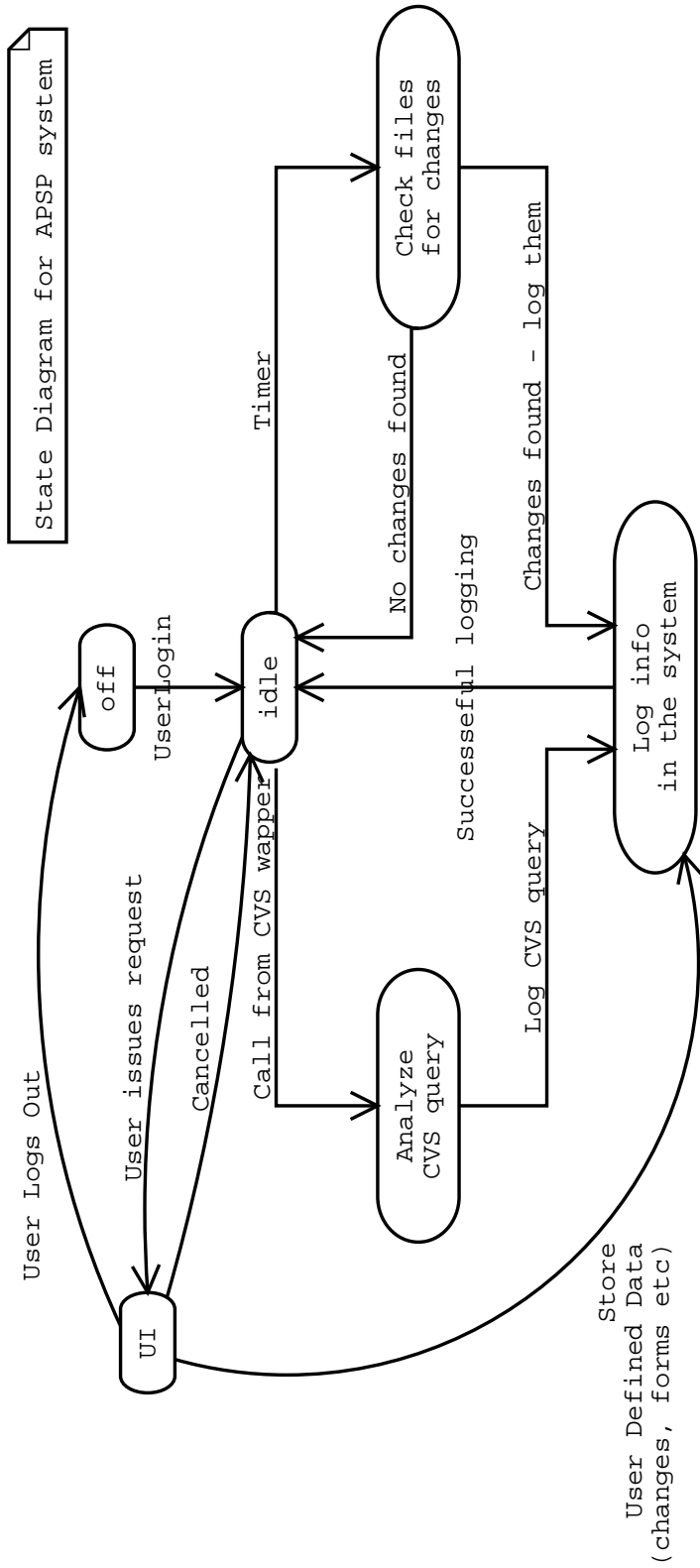


Figure 2.2: State Diagram for APSP System

2.4 Non-functional requirements

2.4.1 Platform

The system will be developed and tested on the Debian 2.4 Linux distribution and hardware capable of running this would suffice.

2.4.2 Dependencies

Out system would depend on whether user uses CVS repository to store his projects' data. So there is almost no sense in out system without running CVS client on the computer.

Other aspects of implementation (choose of database solution etc) will introduce other dependencies, which will be mentioned in product's requisites later.

2.4.3 Security

Security issues shouldn't be under consideration in this stage of the project. Minor security should be done with help of implementation language's standard capabilities.

Contents

1	Business Case	1
2	Requirements	2
2.1	Introduction	2
2.2	Use Cases	2
2.2.1	Use Case “ SystemInit ”: Startup and Initialization of System	2
2.2.2	Use Case “ RegisterAccount ”: Register an account	4
2.2.3	Use Case “ UserAuth ”: User authentication	5
2.2.4	Use Case “ LogOut ”: User Logs Out	6
2.2.5	Use Case “ DeleteAccount ”: Delete user account	7
2.2.6	Use Case “ ChangePassword ”: Change of User’s Password	8
2.2.7	Use Case “ ForgotPassword ”: User forgot his password	10
2.2.8	Use Case “ FillDRC ”: User completes the Design Review Checklist.	12
2.2.9	Use Case “ SwitchPhase ”: User switches between project phases.	13
2.2.10	Use Case “ AddEditPhaseData ”: Add/Edit project phase data.	14
2.2.11	Use Case “ SubmitDefects ”: User submits Defect recording Log (DRL)	15
2.2.12	Use Case “ FillPPS ”: User completes the Project Plan Summary (PPS) form.	16
2.2.13	Use Case “ FillPIP ”: User submits the Process Improvement Proposal (PIP) form.	17
2.2.14	Use Case “ ViewStat ”: User views Graphs.	18
2.3	Domain Model	19
2.3.1	Classes and Associations	19
2.3.2	Queries	20
2.3.3	Actions	22
2.3.4	Events	24
2.3.5	State Transitions	25
2.4	Non-functional requirements	27
2.4.1	Platform	27
2.4.2	Dependencies	27
2.4.3	Security	27

References

- [1] Resources. Psp resources page at the university of karlsruhe. URL <http://www.ipd.ira.uka.de/PSP/>.
- [2] Watt S.Humphrey. *A Discipline for Software Engineering*. 1997.