ijsfontein

Pazzanistraat 10
Westergasfabriek
1014 DB Amsterdam
T. +31 (0)20 420 07 43
F. +31 (0)20 421 45 98
www.ijsfontein.nl

Functional specifications

Cyberdam 2.2

Place	Amsterdam
Date	2 July 2008
Author	Marc de Boer
Position	Technical Manager
Document	Functional specification Cyberdam 2.2 version 0.3.doc
Version	0.3

Table of contents

1 Introduction

This document describes the additions and changes to Cyberdam 2.1 that will result in version 2.2.

The content here is a further specification of the document 'Cyberdam Requirements 2.2', by Pieter van der Hijden [requirements].

1.1 Change history

0.1: Initial draft

2 Improvements in the user interface

The user interface needs a general review to achieve a consistent, practical whole. This will be initiated with a half-day workshop, and followed by a time-boxed period for implementing the most important conclusion.

The items suggested for critical review are described in [requirements].

3 Compliant to educational and technical standards

The exchanging of data between educational systems uses XML files with meta data. To make this possible we add fields to the model and playgrounds and an export function.

3.1 Add meta data to game model and playgrounds

Field	Model	Playground
Name	Existing	Existing
Caption	Existing	Existing
URI (80)	New	Existing
Language (2)	New	New
Description (200)	New	New
Keywords (Large text)	New	New
Contribution (200)	New	New
Size (10)	New	New
Background	Existing 'description' renamed	Existing 'description' renamed
Status	Existing	Existing

New fields are not mandatory. For positioning and more details, check [requirements]. There is no formal check on the values, since they are not intended for use inside Cyberdam, but only for export (see below).

3.2 Export

3.2.1 Template and editing

Add system parameters called 'model meta data export template' and 'playground meta data export template'. They should be edited on a separate screen, so a line with a link on the system parameters page, and be a large text field. Moet dit te editen zijn vanuit de web omgeving? Waarvoor?

For insertion of values in the template we will use the new system for the world model described later in this document. There are two situations where plain insert is not enough: a date that needs formatting and a mapping of the status names to external names, for which specific functions will need to be programmed.

3.2.2 Export function

Add a button 'export metadata' to the playground list and the model list pages. This should export a single XML document merging the tempalate and the meta data, in the same way as other exports work. The output is a single UTF-8 stream. Wat is meta data?

3.3 Integration with Moodle sites

The Cyberdam project website (a separate informational site, not an IJsfontein project) is built using Moodle. It could be possible to do a single identity and/or single sign-on between the website and the games. We reserve 2 days work to investigate the possibilities.

4 Enhancements to maintenance

4.1 Read only access for editors

It happens that people unintentionally modify data. For convenience we add a view function to the major parts. Add a 'view' button next to the 'edit' button on the:

- playground object list page (note: not the playground list page)
- game model list page
- activity list page
- manifest list page

The view button does the same as the edit page except that all fields are read only. To make the implementation practical we don't change the actual HTML elements, but set them to read only and remove some of them. In concreto:

- Input fields (type 'text) get the property 'readonly'
- Select and checkbox elements get the property 'disabled'
- Anchors will lose the 'href' property
- Rich edit fields are replaced by the rich content as HTML, so the whole editor removed.

For everything to still be navigational, some things have to still work:

- Playground object, activities and manifest:
  - 'Cancel' button now has the value ‘Close’ and works the same as before
- Game model:
  - At every step, the 'next', 'previous' and 'cancel' buttons stay enabled
  - In the attachments screen the 'view' link stays enabled
  - Wat gebeurt er met de activities?

The view option is available to all that already have the edit option. In that sense it does not add functionality or allow anything new.

4.2 Read only access to sessions

When sessions have ended (status complete or aborted) participants can still do activities. We will change this so that once the status is complete or aborted, changes can no longer be made. This will be similar to the read only options described above. The link to the session home will stay the same, but all the edit fields and buttons are disabled except:

- The entire menu
- On the home page:
  - the 'view message' link
  - the 'open attachment' link
  - the 'ok' button on messages
- On the messages page:

- the 'view message' link
- the link to the deleted messages

- On the attachments page:
  - the 'open' link
  - the link to the deleted attachments

4.3 Deleting sessions

Currently, sessions are never really deleted. Deleting a non-started sessions results in status 'canceled' and a started session is marked 'aborted'. We want a true delete, keeping the message 'are you sure' in place.

The delete should really permanently remove the session data including attachments from the server.

4.4 Language pack additions

4.4.1 Edit pack page

Add an 'edit' button to the language packs page. It opens a new screen with two fields: code (‘locale’) and description,. Both can be changed. The code has to be unique, the description not. Both are mandatory. OK and cancel buttons as usual.

Note that changing a code could result in invalid user preferences resulting in language-less situation, or inheritance being broken. It is up to the administrator to keep the settings consistent.

4.4.2 Add pack page

A button 'add pack' on the language packs page. Shows the edit screen, but all fields empty.

4.4.3 Delete pack

Add 'delete' to every line of the packs page. On click, show 'are you sure' dialog and on confirm remove pack and all keys and values permanently. Just as with editing, the administrator must be careful of languages used in settings or inheritance.

4.4.4 Inheritance of keys

In some situations it is handy to have partial translations, for example changing some texts to match a more expert audience. For this, we want to have a concept of inheritance in language packs.

Add an optional field 'parent' to every pack, which on the edit screen is a pull down allowing selection of any existing language code, except its own or its children. On saving changes, an extra check for cyclic references will be made.

Whenever a value is needed in the application, the key is first looked up in the language

pack from the user's preferences. If not found, it is search in the parent language pack, and so on. When no match is found, the key itself is displayed in brackets '(' and ')'.

4.4.5 Export

Add an 'export' button to every line of the table with language packs. It produces an XML file with all the data and offers it to the browser for download. The file should be named after the code, so for example 'en.xml'. The file should contain all fields of the pack itself and all keys in the pack. The contents should be in proper readable format using UTF-8 encoding so that the file could be edited outside of Cyberdam. This will require CDATA sections for the rich text fields.

4.4.6 Import

On the language pack list page, add a button 'import' that opens a dialog similar to the playground import page. It should allow browsing to any file.

The import function uses the language code as unique identifier. If a pack with the same key exists, the user sees a 'are you sure' screen to warn about overwriting the data. If confirmed, the existing pack is completely removed and the new data imported. On the summary screen, the number of imported objects is shown.

If the file cannot be completely read for some reason (for example parsing of XML fails) then no data should be lost. Specifically, an overwritten pack should only be discarded after a successful import.

5 Improved playground functionality

5.1 Playground window

In the current implementation, the playground information is accessible through three menu options (per playground) in the session. While reading and writing messages or doing other work in the game, or when composing games, you can't see the map for reference. And when viewing the map, it must be closed to see detailed information on objects. It would be beneficial to see both at the same time.

We will introduce a new window for this. The window can be opened using a link in the header. What playgrounds are visible depends on the user rights. Participants can open the window once a session is opened (because the link in the header is only visible once inside a session) and will see the playgrounds used in that session. All other users will see all playgrounds at any time. Note that if there are many playgrounds, the user might need to scroll to see all of the menu.

The window looks very similar to the way playgrounds are shown in the session now: menu on the left and main view to the right. The menu contains exactly the same as in the session: an entry per playground that can be opened up to show three sub entries: map, address book and description. The options also work the same.

The window opens with a default size of 800x800 which fits the current map nicely, and positioned at 100x100. It is freely resizable and movable.

5.2 Categories per playground

Each playground object falls in a category (industry, government, etc.). We have one set of categories now, but since we will be using playgrounds in a bigger context this is not sufficient. We might use maps in the form of organizational charts at some point, leading to a totally different classification.

Change the following:

- Add a table 'categories' with fields code, description and playground id
- On the change playground screen, add link 'change categories'
- The link opens a new 'categories' screen, with the categories of this playground listed as usual. This means:
    - Header with title, breadcrumb, etc.
    - Introduction text
    - Table with columns id and description
    - Each row has links 'modify' and 'delete'
    - Below the table a button 'add'
    - Normal paging system

    - 'Back' button
- Add or modify happens in a new screen 'category' with
    - fields:
        - code (modifiable!)
        - description
    - Buttons 'save' and 'cancel'
    - Validation that all fields are filled and the code is unique within this playground
- On the screen 'playground object' the category selection is now from the set from this playground
- The multi language keys for the categories are no longer needed: the playground itself is already language dependent.
- The feature is accessible by LCMS administrators only.
- In the whole application, only the description is shown. The code is only used for the ‘type’ element in the map XML, so that the maps can show the proper icons for objects.

6 Game world model

To allow for more mature games, we want to introduce the concept of a world model that exists above the session and participant level. The elements in this world are related to roles, games or even exist independent of these. This could be a concept like money owned by a person or the interest rate in a financial game.

6.1 Script language

In a very wide sense we are starting on a programming language, or more specifically a script language. That is a very big wheel to invent, and we don't want to be building engines indefinitely if this turns into a success.

We will model our language on ECMA script, meaning we use this proven and complete standard as a reference.

6.2 Variables

The world manifests itself as variables with values. The built-in variables are effectively constants: their value cannot change, but we use the same name for consistency.

6.2.1 Variable names

An variable name must start with a letter, underscore (_), or dollar sign ($); subsequent characters can also be digits (0-9). Letters are "A" through "Z" (uppercase) and "a" through "z" (lowercase). Names are case sensitive, and have a maximum length of 255 positions.

6.2.2 Data types

There are no strict data types: all variables are objects. There is one special value 'null', conceptually meaning no value, but is in fact a value in itself.

Objects can have properties, which are also variables, which can be referenced using the dot (.) operator. So the variable 'name' of the object 'user' can be accessed with 'user.name'. For now we don't allow for the definition of objects by users.

6.2.3 Scope

There are three scopes for variables:

- System wide
- Game model
- Role, inside a model

There can be variables with the same name in different scopes. When using the variables they will evaluate in the reverse order above: if a role and a game variable with the same name exist, only the role will be visible.

6.2.4 Predefined variables

Some variables are predefined. They receive values from the system and cannot be changed. Assignments to them will simply have no effect. It is possible, though unwise, to create an variable with the same name, effectively hiding the predefined variable.

The predefined variables are:

- One on each scope level, called 'system', 'model' and 'role'.
- System has an variable 'datetime' which is the timestamp from the server
- Model has the variables defined in its metadata, see above.
- Role has a single variable 'name'

6.3 Defining variables

6.3.1 System level

In the system administrators menu, add a new option 'variables'. This shows a page with the elements listed as usual:

- Header with title, breadcrumb, etc.
- Introduction text
- Table with all fields as columns: name, initialize yes/no and initial value
- Each row has links 'edit' and 'delete'
- Below the table a button 'add'
- Normal paging system
- 'Back' button

Clicking the add or edit button opens an edit screen with:

- name
- checkbox whether there is an initial value. If not, the variable is initialized with 'null' by the system.
- Initial value (only enabled when then the box above is checked): value to initialize with. This cannot be or contain a reference to another variable, it is just a literal value. If “3+4” is entered, the variable value will be “3+4” and not “7”
- Buttons 'save' and 'cancel'

Names are mandatory and validity of the name is checked.

6.3.2 Game model level

Just as above, an option is added to the game authoring menu, which does exactly the same.

6.3.3 Role level

On the role definition page, a link 'variables' is added to every line, with the same functionality as above.

Game model and role variables will be exported and imported with the model.

6.4 Form activities

To allow participants to give values to variables, we introduce a new type of activity: the form activity.

6.4.1 Defining

Add a button to the activities screen, and an edit screen with these fields:

- Name
- Instructions
- Variables. A initially empty list with an 'add' button. Items in the list have a 'remove' and an 'edit' link. The add/edit screen contains fields:
    - Caption. A short text to be displayed next to the input field.
    - Variable. A dropdown contains all variables in scope and a 'select one' option that is initially selected.
    - Mandatory. A checkbox whether the participant should enter a value.
    - Save and cancel

  Caption and variable is mandatory. Variables can only be used once per activity
- Attachments
- Save and cancel buttons

Name, instructions and attachments work as with message activities.

6.4.2 Using

When performing an activity the name, instructions and attachments come up as usual, and the variables are displayed in a table-form, with caption and edit fields next to each other. All input fields are single-line and fixed width, and initially contain their current value. If the value is 'null' the field is shown empty. It is not possible to tell initial values of null and an empty text apart.

When the user leaves the field empty, this is considered an empty string and will not be sufficient if the field is mandatory.

Beneath are cancel and submit buttons, and on submitting the mandatory fields are checked.

Just as with initial values answers cannot be or contain a reference to other variables, only literal values.

6.5 Displaying variables

In rich text fields that the game author specifies, we introduce the possibility of displaying variable values. Anything between an open tag “<%=“ and a closing tag “%>” is taken out of the text, interpreted as an variable name and replaced by the value of that variable.

If the text is not recognized as a variable name, an error is displayed in the text.

Note that this requires us to change the existing template system used for e-mails and the pager. It will have to be decided what to do with existing data: convert on upgrade, leave both systems in place or have the administrator change the data by hand.