## SCHEDULE 2: THE REQUIREMENTS

**All functional assumes the use of CIDOC CRM as the main ontology schema for data**
**Components should be documented**

**Iteration 1 – Semantic Narratives**
**Iteration completion date: June 30, 2017**

| | Title | Description | Contract Status | Notes | Est. |
|---|---|---|---|---|---|
| F1 | Replacement of Draft.JS with Slate | A direct replacement which preserves current functionality (and new mini-templates system) in Semantic Narratives and Annotations. This would allow more options for standard features such as tables but also allow more scope for additional custom requirements. | Fixed | | 12 |
| F2 | Citations in Semantic narratives | The ability to embed Citations via footnotes in the Text Editor system | Optional | Footnotes possibly via Slate plugin | 2 |
| F3 | Charts –ability to embed charts from the search system into a semantic narrative | The ability to embed an interactive chart into the semantic narrative from the clipboard and use it in the same way as it can be used in on the search page.<br>• It should be printable<br>• The dataset underneath (result of the SPARQL query which is fed into the chart) is exportable.<br><br>This should be created as generically as possible so it could be applied (as additional work) to other dynamic objects. e.g. timeline<br><br>The chart from search should be an item available from the clipboard that can be used in a semantic narrative.<br>• Save charts from search component<br>• Persist chart configuration, including the query, dimensions and chart type. But not the visualization state of the chart, like filtered values<br>• Reference/embed chart from UDP | Fixed | | 12 |
| F4 | Ontodia | Integration of Ontodia as library / component for visualisation and | Fixed | | 7 |

| | | | | | |
|---|---|---|---|---|---|
| | | creation of diagrams of CIDOC-CRM ontology and instance data<br><br>The diagram should be an item available from the clipboard that can be used in a semantic narrative.<br><br>Save diagram from Ontodia component<br>Persist component configuration<br>Reference/embed diagram from UDP | | | |

**Iteration 2 – Clipboard, Forms**
**Iteration Completion Date: July 31, 2017**

| | Title | Description | Contract Status | Notes | Est |
|---|---|---|---|---|---|
| F5 | Set Management view | Ability to use existing set-management component in the Set template to view and manage specific set. | Fixed | | 3 |
| F6 | Lightbox / Clipboard Interaction | Replace content of the Related Images tab with what is currently called Lightbox:<br>o implement table selection based on the new event mechanism<br>o update existing table actions to work with event-based selection<br>o Include two faceting options, based on data from 1.) related objects, and 2.) image metadata<br><br>Make functionality that was available in the Lightbox also available in the Set Management view:<br>o Selection of multiple items<br>o Actions on selected items<br>o Image specific actions such as side by side view and overlay view | Fixed | Some functions may be implemented if appropriate and within the same timeframe using the latest stable Mirador version which should also be updated to the latest version if stable. | 5 |
| F7 | Nesting forms | The ability to - within an existing form - define sub groups of fields that belong to another entity but which are still derived from the main entity.<br><br>Currently we use sub forms and the sub form must be saved and then referenced in the main form - to 'connect' it. Until that time it is orphaned but has a URI that is derived from the main form.<br><br>In the new system the user would be able to:<br><br>1. define fields for other domain nodes in the same form.<br>2. not have to manually connect the relationship between the main domain and the sub domain.<br>3. make it look to the user seamless.<br>4. Still be able to surface the same | Fixed | | 10 |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
|  |  | domain information in the sub form as you can in the main form. e.g. display the URI etc. |  |  |  |
| F8 | FORTH Support | Continued support of the technical design, implementation and integration into the ResearchSpace platform of the alignment components developed by FORTH. | Fixed |  | 5 |

## 1 Additional Notes on the Development Approach

ResearchSpace development is split into front-end and back-end development. The interaction between the two is governed by the metaphacts platform and specifically its APIs and Widget Architecture.

### 1.1 Back-End Development

Back-end development will be carried out in the Java programming language, according to the official Oracle Java 8 release. All code will be tested using Oracle Java 8 on the Ubuntu 14.04 LTS release.

The deliverables will be provided with:

- Javadoc-based basic documentation
- Automated build scripts (e.g. Maven, sbt), including test scripting;
- Junit tests

Communication with front-end functionality will be via stateless and HTTP-based RESTful APIs, exchanging either JSON and/or RDF data. Ideally JSON-LD will be used. All APIs will be documented with reference made to test cases and subject to documentation requirements below.

### 1.2 Front-End Development

Front-end development will be carried out in HTML-based Javascript. All code will be tested using the latest version of Chrome and Mozilla Firefox officially released at the delivery date on Windows 7+, Mac OSX 10 and Ubuntu 14.04. The software should also be tested against Ubuntu 16.04.

Javascript is statically typed through the use of TypeScript.

Front-end code is built using React. Front-end developments are packaged using Webpack. Front-end development uses HTML5 features as preference where needed, be tested for validity with the W3C tools. It is assumed that front end code will include appropriate user messages and information (e.g. when waiting, or saving, for conformations, etc