

Deliverable D.1.1 SSP framework

Many collaborative activities have taken place in 2012 that all were aimed at a proof-of-concept portal. The NewProt-internal SSP document is added as an appendix to this Deliverable text. The SSP can be accessed through the internet already, but as the present version is essentially nerd-friendly at best, we will not make it publicly accessible yet. The username and password for access to the portal will be made know to the Commission in parallel with the uploading of this Deliverable text.

D 1.1 SSP Framework  
*NewProt Platform Requirements and Design Specification*

Authors: Andreas Schwarte (FLUID)

Peter Haase (FLUID)  
Michael Schmidt (FLUID)

Contributors: Henk-Jan Joosten (BIOP)

Elmar Krieger (YAS)  
Jiri Damborsky (ENTIS)

Jaroslav Bendl (ENTIS)

Gert Vriend (CMBI)

Co-ordinating institution: fluid Operations AG (FLUID)

**Abstract**

The Self Service Portal (SSP) is intended to provide a platform that will give users access to a broad range of protein engineering tools and services. On an embedded website users can interactively work with these tools and perform their computations supported by integrated workflows. The Self Service Portal provides a tight integration of standard tools for protein engineering and workflow development.

This document is used to specify requirements as well as the system architecture and application design of the SSP.

Requirements are gathered following follows IEEE 830. The requirement specification includes a set of [use cases](http://en.wikipedia.org/wiki/Use_case) that describe all the interactions the users will have with the software. In addition to use cases, the specification also contains non-functional requirements (such as [performance engineering](http://en.wikipedia.org/wiki/Performance_engineering) requirements, [quality](http://en.wikipedia.org/wiki/Quality_(business)) standards, or design constraints).

The design specification follows IEEE 1016-1998, also known as the Recommended Practice for Software Design Descriptions, an [IEEE](http://en.wikipedia.org/wiki/IEEE) standard that specifies an organizational structure for a software design description.

The SSP has been implemented in a first prototype that is accessible for internal use.

Keyword list: *requirements, design, self-service portal, protein engineering*

|  |  |
| --- | --- |
| Document Identifier: | NewProt/2012/D1.1 /v1.5 |
| Class Deliverable: | NewProt KBBE.2011.3.6-01 |
| Version: | v1.5 |
| Date: | November 30, 2012 |
| State: | Final |
| Distribution: | Public |

# NewProt Consortium

This document is part of a research project funded by the KBBE Programme of the Commission of the European Communities as project number KBBE.2011.3.6-01.

The following partners are involved in the project:

|  |  |
| --- | --- |
| **Radboud University Nijmegen (CMBI) – Coordinator**  Netherlands  Contact person: Gert Vriend  E-mail address: vriend@cmbi.ru.nl | **Ernst-Moritz-Arndt-Universität**  **Greifswald (EMAUG)**  Germany  Contact person: Uwe Bornscheuer  E-mail address: bornsche@uni-greifswald.de |
| **BioProdict BV (BIOP)**  Netherlands  Contact person: Henk-Jan Joosten  E-mail address: info@bio-prodict.nl | **Enzymicals AG (ENZYM)**  Germany  Contact person: Ulf Menyes  E-mail address: info@enzymicals.com |
| **Yasara Biosciences GmbH (YAS)**  Austria  Contact person: Elmar Krieger  E-mail address: support@yasara.org | **Ingenza Ltd (INGEN)**  UK  Contact person: Ian Archer  E-mail address: sarah.graham@ingenza.com |
| **S.A.F.A.N. Bioinformatics (SAFAN)**  Italy  Contact person: Luisa Pugliese  E-mail address: info@safan-bioinformatics.it | **fluid Operations AG (FLUID)**  Germany  Contact person: Peter Haase  E-mail address: peter.haase@fluidops.com |
| **LeadPharma Holding (LEAD)**  Netherlands  Contact person: Anna Karawajczyk  E-mail address: | **Enantis s.r.o. (ENTIS)**  Czech republic  Contact person: Zbynek Prokop  E-mail address: |

# Table of Contents

[NewProt Consortium 2](#_Toc343510732)

[Table of Contents 4](#_Toc343510733)

[1. Introduction 5](#_Toc343510734)

[1.1 Background and Motivation for the SSP 5](#_Toc343510735)

[1.2 Methodology and Structure of the document 5](#_Toc343510736)

[1.3 Prototype of the Self-Service Portal 5](#_Toc343510737)

[2. Overall Description 6](#_Toc343510738)

[2.1 Product perspective and Existing relevant protein engineering tools 6](#_Toc343510739)

[2.1.1 MRS 6](#_Toc343510740)

[2.1.2 HotSpot Wizard 6](#_Toc343510741)

[2.1.3 YASARA 6](#_Toc343510742)

[2.1.4 3DM 6](#_Toc343510743)

[2.1.5 HOPE 6](#_Toc343510744)

[2.2 Selection of Use Cases for the proposed system 7](#_Toc343510745)

[3 Specific requirements 14](#_Toc343510746)

[3.1 External interface requirements 14](#_Toc343510747)

[3.2 Functional requirements 14](#_Toc343510748)

[3.3 Performance requirements 15](#_Toc343510749)

[3.4 Design constraints 16](#_Toc343510750)

[3.5 Logical database requirement 16](#_Toc343510751)

[3.6 Software System attributes 16](#_Toc343510752)

[4. System Architectural Design 17](#_Toc343510753)

[4.1 Data Model 17](#_Toc343510754)

[4.2 NewProt Format (NPF) 18](#_Toc343510755)

[4.3 Chosen System Architecture 19](#_Toc343510756)

[4.4 Extensibility 20](#_Toc343510757)

[4.5 Discussion of Alternative Designs 20](#_Toc343510758)

[4.6 System Interface Description 20](#_Toc343510759)

[5. Detail Description of Components 21](#_Toc343510760)

[5.1 MRS 21](#_Toc343510761)

[5.2 HotSpot Wizard 21](#_Toc343510762)

[5.2.1 HotSpot Wizard Communication Protocol 21](#_Toc343510763)

[5.3 YASARA 22](#_Toc343510764)

[5.3.1 YASARA Server 22](#_Toc343510765)

[5.3.2 YASARA Client 22](#_Toc343510766)

[6. USER INTERFACE DESIGN 23](#_Toc343510767)

[6.1 Description of the User Interface 23](#_Toc343510768)

[7. Roadmap and Next Steps 27](#_Toc343510769)

[Appendix 28](#_Toc343510770)

[I. NewProt Format Schema definition 28](#_Toc343510771)

# 1. Introduction

## 1.1 Background and Motivation for the SSP

Researchers and practitioners working on protein engineering can select from a broad variety of tools and services to address common workflows. For many of these workflows several tools are required with the consequence that intermediate results have to be passed between the different tools with manual effort. Although there exist standards for data formats as well as interfaces for exchange of intermediate results, a tight integration of the different tools supporting practical use-cases is missing.

The Self Service Portal (SSP) will provide a platform that will give users access to a broad range of tools and services. On an embedded website users will interactively work with these tools and perform their computations supported by integrated workflows. The Self Service Portal will provide a tight integration of standard tools for protein engineering and workflow development. Throughout the workflows intermediate results can be retrieved and visualized in a unified user interface.

The SSP is designed based on the open source Information Workbench from fluidOps. The SSP will provide a working environment to easily interact with all produced NewProt resources. To this end, the SSP will enable database access (for WP2), software access (for WP3), system facilities (SVN access, ftp access, virtual machine, etc.), dissemination (e.g. course material, software documentation) help facilities, and project management facilities. Users will be able to interactively run NewProt software within the SSP workbench.

Software and databases will be fully interoperable, i.e., users will not need to store in-between results and will not need to worry about file formats etc. This interoperability requires that all data types will be syntactically and semantically described (WP2) and that all software that can operate on those data will need to 'know' about the syntactic and semantic annotations (WP3-5). The use of common standards (RDF, Linked Data, as well as relevant domain ontologies such as EDAM, OBO and the GO Gene Ontology)) will ensure the semantic interoperability of data and software. Much of the data and software the partners intend to incorporate in the SSP already use the SOAP protocol for interoperability and adhere to these common standards and ontologies.

## 1.2 Methodology and Structure of the document

We follow standard guidelines from software engineering for requirements analysis and design specification. In particular, we create a software requirements specification (SRS) and software design description (SDD).

A requirements specification for a software system is a complete description of the behaviour of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints). The requirement specification follows IEEE 830.

A software design description (SDD) is a document used to specify system architecture and application design in a software related project. For the Software Design Descriptions we follow IEEE 1016-1998, an IEEE standard that specifies an organizational structure for a software design description.

## 1.3 Prototype of the Self-Service Portal

A first prototype of the SSP as described in this document has been implemented. This prototype is accessible on the web at <http://newprot.fluidops.net/>. The prototype is intended for project internal purposed (showcasing, validation and evaluation) with the following credentials:

User: newprot Password: newprot

# 2. Overall Description

## 2.1 Product perspective and Existing relevant protein engineering tools

The starting point for the Self-Service Portal is a set of existing tools that are used by protein engineers and researchers today, albeit with limited interoperability between them. We provide a brief description of the relevant tools and their purpose:

* MRS
* HotSpot Wizard
* YASARA
* 3DM
* HOPE

### 2.1.1 MRS

MRS is one of the NewProt resources to be integrated into the SSP. MRS is a search engine for biological and medical databanks and provides fast access to over a terabyte of indexed data from all standard databanks such as Uniprot or Genbank. In the SSP the MRS interface will be integrated as means to search for proteins and to retrieve information. The integration of MRS will be done on exposed web services. With such integration the user does not have to leave the SSP platform, and still has all resources and services at hand.

### 2.1.2 HotSpot Wizard

The HotSpot Wizard is a computational service to automatically compute so-called *hot spots* in protein structures. These *hot spots* can be used for the processes in protein engineering, as well as for annotation of protein structures. As output the HotSpot Wizard provides lists of annotated residues including estimations for mutability.

In the SSP the Hotspot Wizard will be integrated as a web application. From the portal, users can invoke computations on proteins using the tool by passing PDB files to the HotSpot Wizard service. Once the computations are finished, the results (i.e. the annotated residues) are accessible from the SSP.

### 2.1.3 YASARA

YASARA is a molecular modeling and simulation program including facilities for visualization of such resources. In the SSP, we employ YASARA on the one hand as a rendering engine for protein structures (running as a local server application), and on the other hand as a client application to visualize prepared YASARA scene descriptions.

### 2.1.4 3DM

3DM is a commercial tool with extended capabilities. 3DM provides protein superfamilies systems based on structural superpositions and multiple sequence alignments. The SSP will link alignments, proteins, and amino acids to corresponding entries in 3DM. If the commercial features are activated in the portal a high-quality 3DM system for the selected protein will become available

### 2.1.5 HOPE

HOPE is an easy-to-use webserver that analyses the structural effects of a mutation of interest. The server allows to submit a protein sequence and the mutation. HOPE will then collect and combine available information from a series of webservers and databases and will produce a mutation report complete with results, figures and animations.  Where available Project HOPE will use the 3D structure of the protein but the server can also build a homology model if necessary. Other information sources include the Uniprot database and a series of DAS prediction servers.

## 2.2 Selection of Use Cases for the proposed system

In this section, we present a set of selected use-cases. These use-cases will illustrate the possibilities for integration and the benefits for users.

|  |  |
| --- | --- |
| Use case 1 | Match a sequence to protein identifier |
| Use case 2 | Finding PDBs associated with a Protein ID |
| Use case 3 | Linking proteins to alignments |
| Use case 4 | Compute model with YASARA |
| Use case 5 | Use Hotspot Wizard output as input for YASARA |
| Use case 6 | Generate HOPE report for a protein |

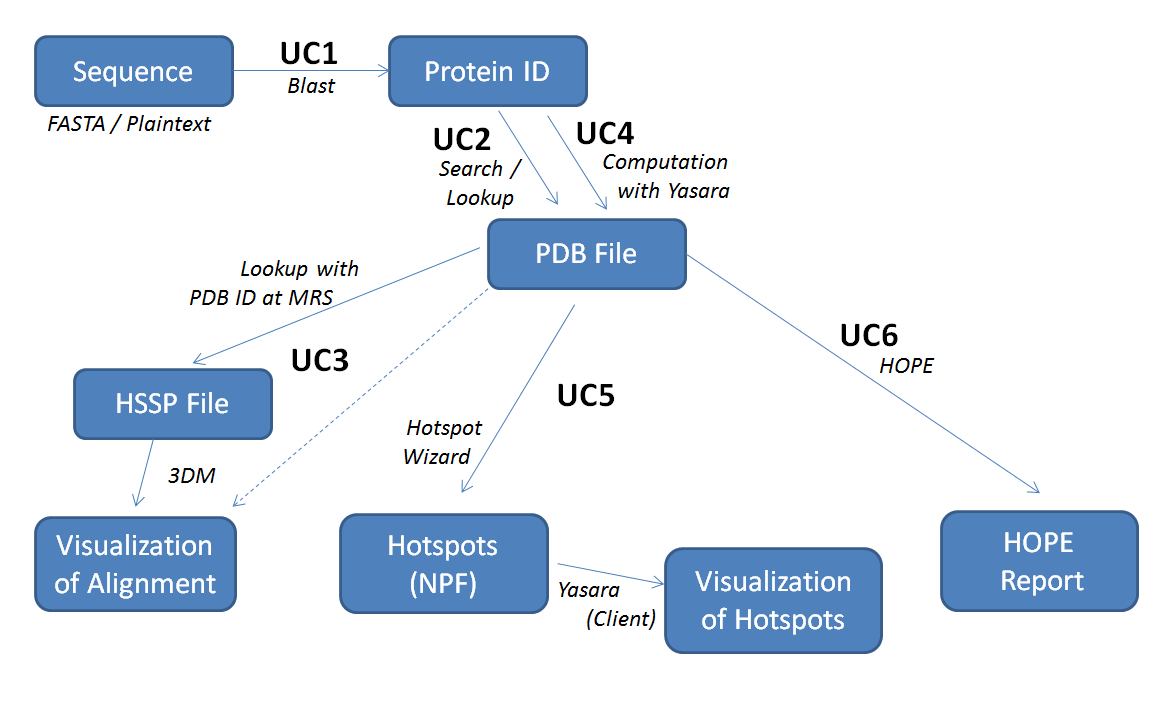
****

Figure 1: Interaction between use cases

|  |  |
| --- | --- |
| **Use case 1: Match a sequence to protein identifier** | |
| Objective | The objective of this use case is to retrieve an existing protein identifier matching the user-provided *FastA* sequence. Each protein in the system is identified by such an ID and additional information (such as for instance models) is associated to it. Proteins in the SSP will be identified uniquely, see Section 4.1 Data Model |
| Stakeholders | This functionality addresses all users of the portal as it provides an entry point for protein lookup. |
| Significance | With the integration of the search interface into the portal, a single point of access is established. Users do not have to employ multiple different services, nor do they have to deal with data exchange between different tools. |
| Methods | 1. The user provides a sequence in *FastA* format (or plain text) using a text field 2. A protein identifier is retrieved using the Blast Service 3. Result is visualized in the portal |
| Components | * MRS Blast Service * NIH (official Blast) for commercial applications |
| Risk | Through the use of external services a dependency on these services is introduced with respect to the quality. This might in particular become a risk for the commercial (closed-source) Blast service from NIH. |

|  |  |
| --- | --- |
| **Use case 2: Finding PDBs associated with a Protein ID** | |
| Objective | The objective of this use case is to search for or lookup an existing PDB file using the given protein identifier. The protein model is maintained in the local database of the NewProt portal. |
| Stakeholders | This functionality addresses all users of the portal as it provides an entry point to retrieve protein models as PDB file, the basis for many use cases. |
| Significance | The protein model represented by a PDB file is the basis for most scenarios in the SSP (in particular UC3, UC5 and UC6). This model is used as input for all protein engineering tools. |
| Methods | 1. If the PDB file is already available in the local database, continue with step 4. 2. Search for PDB file at MRS using the Uniprot ID 3. Store the PDB file in the local database 4. Show a summary page for the given protein, including a link to the PDB file |
| Components | * Local database * MRS |
| Risk | Through the use of external services a dependency on these services is introduced with respect to the quality. |

|  |  |
| --- | --- |
| **Use case 3: Linking proteins to alignments** | |
| Objective | The objective of this use case is to find the alignments for a protein and to visualize the results in the commercial tool 3DM or in another visualization tool. |
| Stakeholders | Any user of the portal |
| Significance | This use case captures the central link between protein and alignments and provides and easier way to activate operations. |
| Methods | 1. Using the PDB ID as input, the HSSP file is looked up within the MRS HSSP database 2. The retrieved HSSP file is stored in the local database 3. A link to 3DM is created based on the PDB ID   *Alternative: If the model is computed (e.g. by YASARA as in UC4), the HSSP needs to be computed as well using a web service.*  Note: 3DM does not actually need the HSSP files, however, they might be required for other visualization tools. |
| Components | * MRS HSSP database * 3DM * Alternative visualization tools (e.g. jalview.org) * HSSP generation web service |
| Risk | A risk might be the identification and the integration of alternative visualization tools |

|  |  |
| --- | --- |
| **Use case 4: Compute model with YASARA** | |
| Objective | The objective of this use case is to compute a protein model using the YASARA structure client. This use case captures the scenario where the model cannot be retrieved from an existing database. |
| Stakeholders | Any user of the portal |
| Significance | This use case complements UC3 with the possibility to compute protein models if the PDB file cannot be retrieved using an existing database. |
| Methods | 1. Using the protein sequence, YASARA or the Protein Model Portal are used to compute the PDB file 2. The PDB file is stored in the local database 3. Show a summary page for the given protein, including a link to the PDB file |
| Components | * YASARA Structure on Server * PMP service (Protein Model Portal) |
| Risk | Computation of protein models from the sequence is a lengthy process. Risks are the computational costs as well as concurrency issues (i.e., parallel computation, management of parallel computation tasks, etc.) |

|  |  |
| --- | --- |
| **Use case 5: Use HotSpot Wizard output as input for YASARA** | |
| Objective | The objective of this use case is to employ the hot spots computed with the HotSpot Wizard as additional input for YASARA. In YASARA these hot spots can be used for further analyses. One goal of this use case is to facilitate protein engineering workflows in terms of data exchange between standard tools, i.e., the intermediate results are passed between the tools automatically. |
| Stakeholders | Any user of the portal |
| Significance | The tight integration of the HotSpot Wizard and YASARA from a user’s perspective, support and simplify standard protein engineering workflows, in particular because intermediate results do not have to be passed manually between the tools |
| Methods | 1. Using a PDB file the HotSpot Wizard computations are invoked 2. The HotSpot Wizard (asynchronously) returns the hotspots in the NewProt Format (NPF) 3. User selects NPF file to trigger visualization in YASARA |
| Components | * HotSpot Wizard * YASARA |
| Risk | A risk might be the computation costs (in particular CPU power) as well as concurrency issues. In addition, an agreement on the data interchange format – the NewProt Format (NPF) – has to be found. |

|  |  |
| --- | --- |
| **Use case 6: Generate HOPE report for a protein** | |
| Objective | The objective of this use case is to generate a HOPE report for a given protein. An example scenario could be the question of which mutation to make for a given protein to achieve stabilization. |
| Stakeholders | Any user of the portal |
| Significance | The integration of this use case into the portal provides a single point of access and simplifies user workflows |
| Methods | 1. Using a PDB file (see UC2/UC4), HOPE is used to build a homology model 2. HOPE results are embedded in the portal associated with the protein (Polling for results) |
| Components | * HOPE |
| Risk | None |

# 3 Specific requirements

## 3.1 External interface requirements

The SSP must provide the following external interfaces:

* End-users can access and use the SSP via a Web Browser, using either HTTP or HTTPS protocols. The system should support the latest versions of the most prominent browsers (in particular the latest versions of Internet Explorer and Firefox). The HTML pages delivered by the SSP are plain HTML5/JS pages, such that the basic functionality can be accessed without installing any external software or browser plugins (e.g., no Flash Player needs to be installed). Note, however, that this restriction does not necessarily hold for external components that are integrated into the SSP.
* Access to data residing inside the SSP is possible through a SPARQL interface, following the official *W3C SPARQL Protocol* for RDF recommendation. Using this protocol, authorized users can extract and export data residing inside the system into several standard serialization formats such as RDF/XML, N3, JSON, etc.
* Administrators can access the SSP via a Command Line Interface (CLI), using remote method invocation (RMI) for communicating with the backend. The CLI provides basic functionality for administrating the system, including system backups and the manipulation of data residing inside the system.

## 3.2 Functional requirements

**F1:** User registration process

* The user must be able to register to the system.

**F2:** Upload and delete PDB Files

* The user must be able to upload/import PDB files into the system.
* PDB files in the system must be deletable by users

**F3**: Communication with MRS

* Search in MRS database from inside the SSP
* Transfer/Interlink PDB files related to MRS search inside the system (similar to F2)

**F4**: PDB file and residue library

* Survey page listing available PDB files, which can be selected for further processing
* Same for residues (which, for instance, could have been generated from HotSpot Wizard)
* Links between PDB files and residues (“provenance” information)

**F5**: Communication with YASARA Server:

* Trigger scripts on YASARA application running on the SSP host
* Option 1: YASARA delivers PNG, must be possible to display inside the SSP
* Option 2: YASARA delivers scene file, which can be displayed with a YASARA client running on the user’s machine (see F6)

**F6**: Communication with YASARA Client:

* Feed YASARA client running on the user’s computer with a PDB file from the SSP
* Feed YASARA client with a scene file from the SSP

**F7**: Communication with 3DM

* Link proteins, amino acids, and alignments to pages in the corresponding 3DM system.

**F8**: Communication with HotSpot Wizard

* Process PDB file from SSP’s PDB library with HotSpot wizard (integrated into the SSP)
* Store the result delivered by the HotSpot Wizard inside the SSP

**F9**: Workflow management from the portal

* Selection of inputs (two files)
* Use external tool to compute output
* Get some value from the generated output
* Display this value (associated to the workflow)
* Email notifications

**F10**: Workflow requiring external computational resources

* Selection of inputs (one file, one folder)
* Use external tool to compute output
* All combinations have to be computed in a loop
* Get some specific value from each generated output
* Take top-k combination with respect to the output value
* Use top-k combinations for further (long computation) mutations => 1month
* Present final outcome
* Email notifications

## 3.3 Performance requirements

We define the response time as the interval between a user-command and the receipt of feedback from the system. In the context of interaction with a Web-based system like the SSP, this means the interval between the user action (such as a click on a link or button) and the delivery of an initial page including feedback (yet not necessarily the final result). The response time of the SSP’s HTML interface shall be less than 2 seconds in 90% of the cases, and less than 10 seconds in 99% of the user requests.

However, this requirement is relaxed to hold for the main navigation pages of the SSP only. For survey pages constructed based on cost-intensive computing steps (such as the aggregation of data from different sources) or external tools embedded into the SSP higher response times can be tolerated.

The SSP shall scale to at least 20 simultaneous users. This means that the response time requirements above shall hold for 20 simultaneous users. Up to 50 simultaneous users, an increase of the number of simultaneous users shall lead to an at most proportional increase in the response time limits defined above.

The SSP shall have an availability of 99.9% (not including announced down-times of at most 3 days per year for maintenance).

The SSP shall have an automatic fault recovery mechanism, which recovers a consistent state of the system in case of a critical error.

## 3.4 Design constraints

This section describes the technology and design constraints for the SSP.

* **Standard Compliance**
  + The HTML delivered by the SSP shall be HTML5-conform
  + The access to the SPARQL interface shall be conform with the W3C’s latest SPARQL protocol specification
  + The SSP must be designed in such a way that it is fully functional using the latest versions of the most important Web browsers.
* **Hardware Constraints**
  + The SSP requires at least 8GB RAM and a quad core 2.00 GHz CPU (or comparable hardware)
  + The SSP shall be installable on Amazon AWS cloud infrastructure as well as private clouds driven by VMware.
* **Software Constraints**
  + The SSP shall be implemented as a platform-independent Java application.
  + The SSP is designed to run on 64bit systems.
* **Other Constraints**
  + The SSP requires Internet connection to communicate with the external protein engineering tools.

## 3.5 Logical database requirement

This section describes the logical database requirements in the SSP.

* Modeling of data model as ontology
* Flexibility for extensions
* Reuse of existing vocabulary
* Integration of relevant existing data (Uniprot)
* Need for standard XML-based interchange format between components

## 3.6 Software System attributes

This section describes the software’s system attributes, including the mandatory attributes (for functional completeness) and the optional attributes for production use in enterprise environments

* **Security**
  + Use of common security standards (HTTPS, SSH, …)
  + Logs & audit trails
  + User authentication mechanism (optionally: RSA-certified)
  + Protection against DoS-attacks (optional, for production use)
* **Availability**
  + Support for active-passive failover configuration (optional, for production use)
  + Load balancing support (optional, for production use)
  + Automatic recovery, backups
* **Maintainability**
  + Developed on standards
  + CLI for maintenance at runtime
* **Portability**
  + Implemented in Java
  + Avoidance of platform-dependent libraries
  + Use of standardized web-technologies
* **Testability**
  + Reached through modularization & code design
* **Usability**
  + Usable by domain experts without deep IT knowledge

# 4. System Architectural Design

In this section, we present the data model and the architecture of the SSP. The portal will be designed based on the open source Information Workbench from *fluid Operations* and will provide a working environment to easily interact with all produced NewProt resources. A design goal of the SSP is to allow for project based collaboration between users.

In the remainder of this section we present the data model of the portal, discuss the chosen architecture, present a set of alternative design options, and finally close with a description of the system interface.

## 4.1 Data Model

A design goal of the SSP is to provide a unified view on the NewProt resources, including computational tools, databases and interfaces. Users will be able to interactively run NewProt software within the SSP workbench, while all Software and databases are fully interoperable, i.e., users do not need to store intermediate results and do not need to worry about file formats etc. To achieve this level of interoperability between different computational tools and services, the data model will be based on standard formats, e.g. PDB signatures for proteins. The overall data model is depicted in Figure 2 and described in detail in the following.

In the portal data is organized in a way that allows collaborative workflows involving multiple users. These workflows are represented by the notion of projects which have a number of associated users - the project owners or contributors. Note in particular that multiple users can work on the same project. Within these projects all associated resources for the project are maintained, i.e. computational workflows and past results.

Projects provide a unified view on a user’s ongoing work: contributors to the project can see all relevant information about the project, in particular the current state, all applied resources, and the computational results. In a project one or more of the NewProt software components are used for performing computations on the proteins defined by workflows. In the data model this is reflected by the 1:N relation of projects to proteins, where one project can use multiple proteins as resources.

To allow for interoperability between the NewProt software components, proteins are represented in standard formats. This is the PDB signature of the protein and the HSSP definition. These representations are maintained in files which are associated to the particular protein.

For visualization of proteins we associate scenes to the PDB representation of a protein. This is either a rendered PNG or a description of the scene in SCE format which can be processed further by other NewProt components. These visualizations are also maintained in the file system and can be displayed in the portal on-demand (associated to a protein).

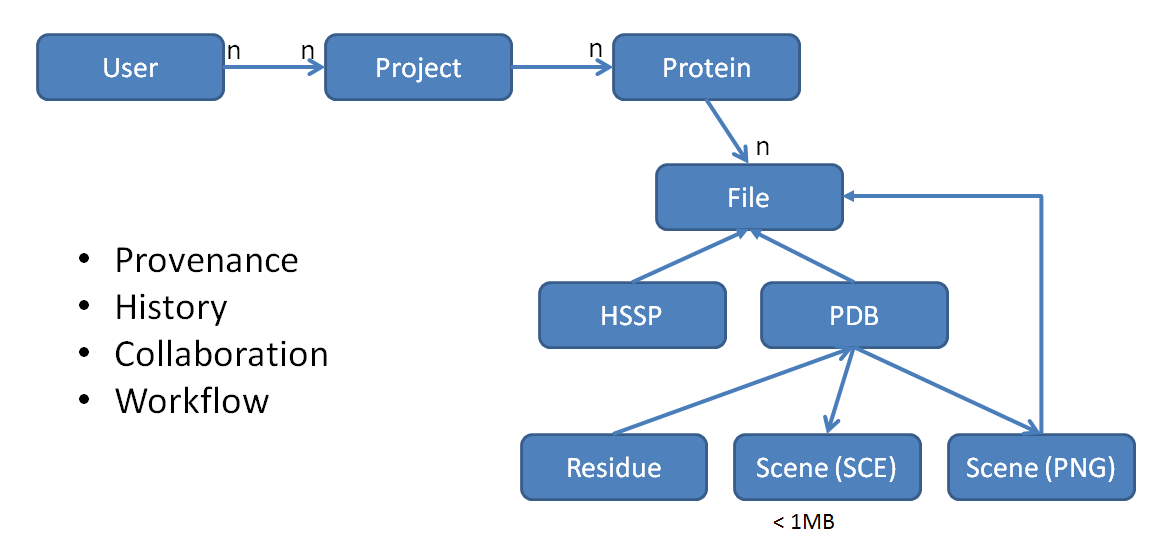


Figure 2 Overview of the SSP data model

In the SSP proteins are uniquely identified[[1]](#footnote-1),

1. by their Uniprot identifier (if the protein exists in Uniprot)
2. by a hashed identifier (with a user label) for custom proteins

Computational tasks in the SSP are modelled by the notion of a Job. Each job is owned by a particular user and links to a particular protein. All jobs can be tracked from a designated job page, see Section 6 for details.

## 4.2 NewProt Format (NPF)

The NewProt Format (NPF) is a XML-based file format for data exchange between the different software tools. The format allows representing residues as a mapping from *Residue* to *Residue Value*. The NewProt Format is relevant for the following software components:

* HotSpot Wizard
* YASARA
* 3DM
* HOPE

The NewProt Format is defined by the *XML Schema Definition* in Appendix I and allows specifying metadata (such as e.g., the source, the title, or the creation date) using the Dublin Core vocabulary[[2]](#footnote-2). In the body of the XML document a list of residues can be represented, each of which is associated with a set of values. The supported values and their format are defined in the schema, and include

* *mut*: Target value expressing the degree of mutability
* *flags*: Flags for residue, i.e. Catalytic, Pocket, Tunnel
* *alignments*: Definition of alignments of this residue

An example residue mapping from the body of an NPF document looks as follows:

<residueValue isHotspot=*"true"*>

<residue>

<number>2</number>

<chain>A</chain>

<type>THR</type>

<pdb\_number>1</pdb\_number>

<insertion\_code></insertion\_code>

<model\_number>1</model\_number>

</residue>

<value>

<mut reliable=*"false"*>0</mut>

<flags pocket=*"true"* catalytic=*"false"* tunnel=*"false"* />

<alignment>

<variant><type>THR</type><count>7</count></variant>

<variant><type>ILE</type><count>1</count></variant>

</alignment>

</value>

</residueValue>

## 4.3 Chosen System Architecture

The Self Service Portal integrates multiple computational tools as well as services relevant for protein engineering. As already discussed, the major design goal of the SSP is to provide the user with a unified view on the NewProt resources. In the following, we describe the chosen architecture for the SSP and discuss the integration of these components.

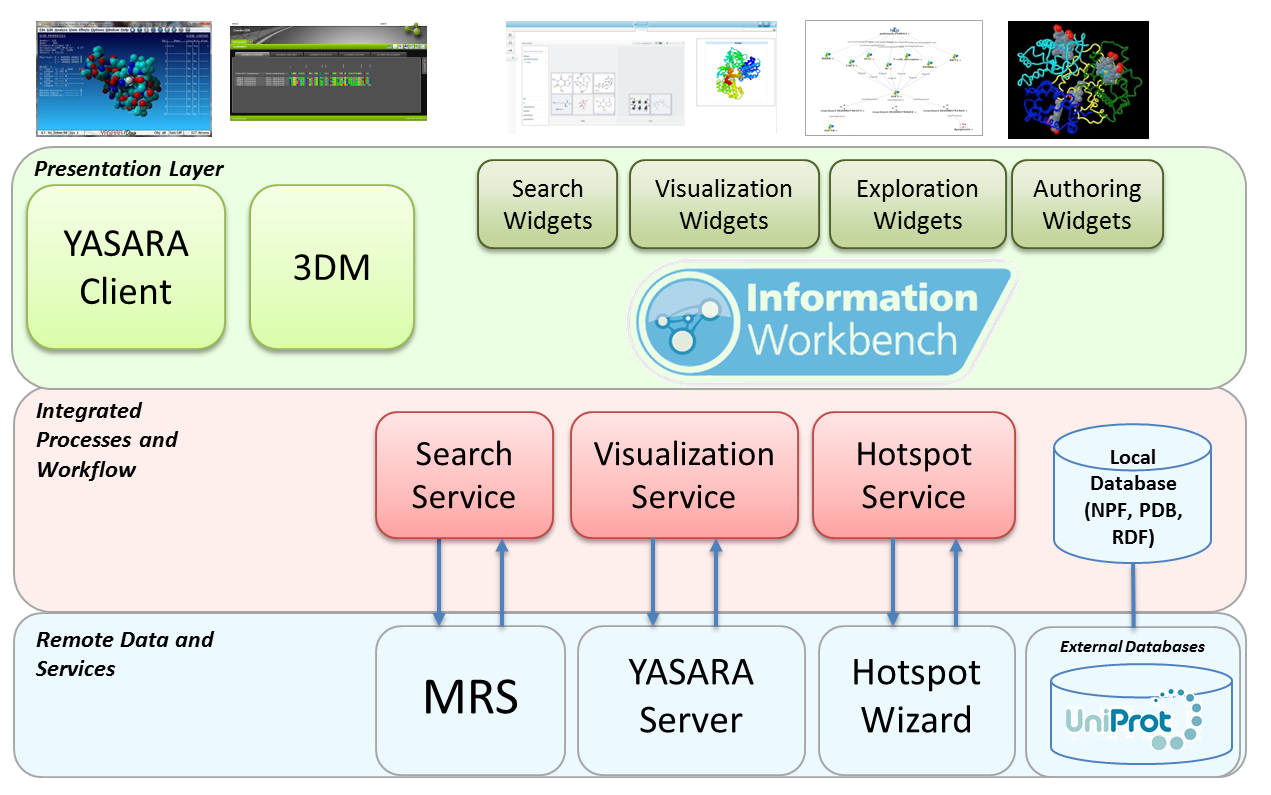


Figure 3: Architecture of the SSP

The architecture of the SSP (see Figure 3) is designed based on the open source edition of the Information Workbench from *fluid Operations*. Serving as a platform for Linked Data, the Information Workbench enables users for collaboration, integration of public as well as private data and services, and analytics on the data. Users benefit from a unified view both on the data level and on the integrated resources.

In the NewProt project the Information Workbench is the layer of integration, providing a unified view of the data and components. The architecture of the SSP is divided in three layers centered around the core platform:

1. *Remote Data and Services*
2. *Integrated Processes and Workflow*
3. *Presentation Layer*

The first layer all NewProt software components are summarized, including the MRS database, a local YASARA application, the HotSpot Wizard as well as external databases. Please refer to section 5 for a detailed description about the individual components and services and how they are integrated into the SSP (e.g. the communication protocols).

The second layer comprises the integrated services of the SSP required for the interaction between the NewProt software components and the Information Workbench. In addition, this layer contains the local database which stores PDB models, NPF files as well as structured data in RDF.

The Presentation Layer is built with the UI components Information Workbench, in particular the *Search*, *Visualization*, *Exploration* and *Authoring* Widgets. In addition, the SSP uses the visualization of the commercial software 3DM as well as the YASARA client.

## 4.4 Extensibility

The Information Workbench as the underlying platform for the SSP is designed with extensibility in mind allowing for the integration of further software components as well as other customizations (e.g. in the user interface). The platform comes with different concepts for the integration, including

* **API Framework**
  + New API services can be plugged into the platform as *dynamic services* and are for instance immediately accessible from the command-line interface (CLI) or the RESTful endpoint
* **Widget Concept**
  + The Information Workbench ships with a set of standard widgets (e.g. for representing tables, charts, graphs, etc.), and can be easily extended with custom, specialized widgets.
* **Provider Concept**
  + Providers are configurable programs to integrate data from remote sources into the local database. The Information Workbench can be extended with custom providers.
* **Data Model**
  + The underlying data model uses an ontology which can be extended with further concepts
  + The NewProt format is defined as XML Schema and allows for adding further attributes, etc.

## 4.5 Discussion of Alternative Designs

For the integration of the different software components there are multiple alternative design decisions to make. In the following, we will briefly summarize the options.

One option is to integrate software components tightly into the platform itself. In this setting 3rd party software becomes part of the platform itself and in particular the user interface is entirely integrated. In contrast, a loose integration offers more flexibility and the software component is integrated for instance via standard web services. This is the paradigm we are using in the SSP.

The next design decision regards the location where external software components are running. Often software components expose their functionality via standard web services. These can be hosted local to the SSP itself (e.g. in the same application server), or on some remote machine from where they are accessed via HTTP over the internet. Different constraints influence a decision on the chosen paradigm, including in particular scalability and quality of service. The selected paradigm might also be changed depending on the installation environment, e.g. in enterprises it might be a requirement that all (or most) external tools are running in the local computing environment. For the initial phase of the NewProt project all software components that are integrated as web services are running on a remote site.

## 4.6 System Interface Description

The Information Workbench as an underlying platform to the SSP provides different system interfaces:

* A web-based interface
  + Browser based access to the SSP via HTTP(S)
  + Support for modern browsers
* API access via CLI and REST
  + Command-Line interface for administrative tasks
  + RESTful interface for 3rd party communication
* NewProt Software components
  + Web technologies (e.g. web services)
  + Application start via Download (e.g. YASARA scene)
  + Socket communication between processes (i.e. YASARA)

# 5. Detail Description of Components

## 5.1 MRS

MRS provides services that allow searching for proteins and structures using large well-defined databases. In the portal we employ these services to search for existing proteins in the Uniprot database. In addition, we use the blast functionality of MRS to find proteins matching a given *FastA* sequence.

In the SSP the MRS service is accessed using standard web service technologies, in particular SOAP. The web service is hosted by *CMBI*[[3]](#footnote-3) and is accessed through the public internet.

## 5.2 HotSpot Wizard

The HotSpot Wizard is a computational tool which amongst other features takes PDB files as input and computes relevant residues for mutational studies that can be used by other tools for further analyses and computations. In the SSP, the HotSpot Wizard is accessed through standard RESTful web interfaces. The HotSpot Wizard services are hosted externally and are accessible via HTTP.

### 5.2.1 HotSpot Wizard Communication Protocol

The communication protocol between the SSP and the HotSpot Wizard is defined as described in the following.

1. The SSP sends a hotspot computation request, passing a reference to the PDB model (optionally some special HSW arguments)  
     
   a) if PDB is available in RCSB PDB[[4]](#footnote-4)

*http://ll06.sci.muni.cz:8080/hswRest/rest/pdbFromId?pdbId=<PDB\_CODE>&caller=<CALLER>&responseUrl=<RESPONSE\_URL>*   
   
where <PDB\_CODE> is four-letter PDB-code, <CALLER> is “newprot” and <RESPONSE\_URL> is the URL for sending result information once the computation is finished

b) PDB retrieved via link

*http://ll06.sci.muni.cz:8080/hswRest/rest/pdbFromLink?link=<STRUCTURE\_LINK>&caller=<CALLER>&responseUrl=<RESPONSE\_URL>*where <STRUCTURE\_LINK> is link to the user's structure in PDB file format after performing URL encoding, <CALLER> is “newprot” and <RESPONSE\_URL> is the URL for sending result information once the computation is finished

1. The HotSpot Wizard service schedules a job and returns the job identifier as an immediate response. The job identifier consists of 50 alphanumeric characters.
2. The SSP manages the asynchronous job locally with its job manager
3. Once the job is finished, the HotSpot Wizard informs the SSP, that the computation result is ready to be picked up using a restful API service. As part of the request a reference to the result file (in NewProt format) is passed as argument.  
     
   a) informing about success  
     
   informHotSpotComputationFinished(String hotspotJobId, String resultFileUrl)  
     
   *http://<NEWPROTPORTAL>/REST/JSON/getNewprotService/?method=informHotSpotComputationFinished&params=[<hotspotJobId>,<resultFileUrl>]&id=1*  
   b) inform about failure  
     
   informHotSpotComputationError(String hotspotJobId, String error)  
     
   http://<NEWPROTPORTAL>/REST/JSON/getNewprotService/?method=informHotSpotComputationError&params*=[<hotspotJobId>,<error>]*&id=1
4. The SSP retrieves the result from the HotSpot Wizard using the given reference, and stores the file locally for further use

## 5.3 YASARA

For YASARA we distinguish between the YASARA server and the YASARA client. The YASARA server will be running as a server process on the Self Service Portal’s host, while the YASARA client is installed on the user’s machine.

### 5.3.1 YASARA Server

The purpose of the YASARA server is to provide means for rendering scene objects (i.e. PNG or SCE scenes) of a selected protein. The YASARA server application is running in its own process which is controlled from a YASARA Java wrapper by the SSP. The YASARA Java wrapper exposes all relevant functionality through a simple Java interface and takes care for all inter-process communication using sockets. For the SSP a YASARA API was defined that wraps up required function (see Figure 4).

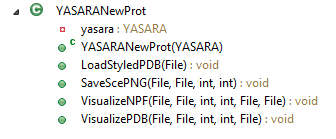


Figure 4: YASARA API for NewProt

Using this YASARA API the location references of the local PDB file is passed to YASARA, which can then perform the desired action, e.g., creating visualization files of hot spots as PNG and YASARA SCE (UC5).

### 5.3.2 YASARA Client

In contrast, the YASARA client is running on the user’s machine providing means for more extensive visualization and further local analyses. From the portal the user can download a pre-computed YASARA scene file and open this with a local YASARA.

# 6. USER INTERFACE DESIGN

In this section we describe the user interface of the SSP.

We provide parts of the user interface design in our prototypical implementation available at http://newprot.fluidops.net[[5]](#footnote-5).

## 6.1 Description of the User Interface

**I1**: Login screen

To interact with the SSP, users are required to authenticate against the system’s credentials database. Hence, when the user first accesses the SSP, a login screen similar to the one depicted in Figure 5 is displayed.

In the login screen the user is asked to enter a valid user name and the corresponding password. After successful login, the user is redirected to the user start page as described below.

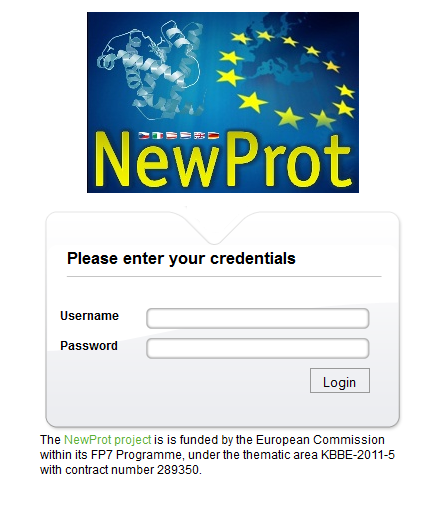


Figure 5: Login screen of the SSP

**I2**: User start page

The user start page provides the entry points to all SSP functionality (see Figure 6). On the left-hand side various entry points for finding proteins are listed, including

1. MRS Blast interface (see Figure 7) – UC1:
   * *FastA* sequences can be blasted against the MRS web service[[6]](#footnote-6)
   * Matches are retrieved ranked by score
   * Users can select a protein from the matches and use it in the SSP
2. Existing proteins
   * A list of proteins already that have been created in the SSP
3. Jobs Page
   * A special page to keep track of running/finished jobs, e.g., for long running hotspots computations (UC5) the status can be looked up on this page
4. A full-text search interface
   * Full-text search for proteins in Uniprot using the MRS database
   * Results are retrieved ranked by score
   * Users can select a protein from the matches and use it in the SSP

On the right-hand side an entry point for each use case is provided, linking to a page with a description and an example of how the particular functionality can be used in the SSP.

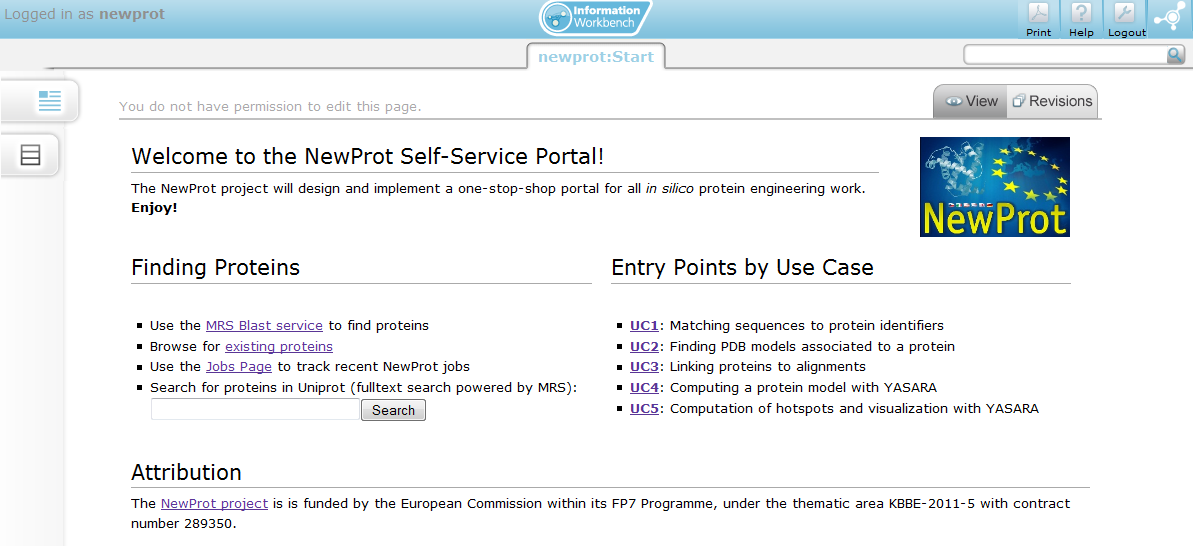


Figure 6: User start page in the SSP

**I3**: Project page

The project page offers the following functionalities to users:

* Track running jobs of the project, e.g., computation of hot spots
* View and manage associated proteins
* Manage associated files (e.g., PDB files)
* Manage project contributors
* Entry points for workflows

The project page is only partially implemented in the prototypical implementation, where most functionality is available from the user start page.

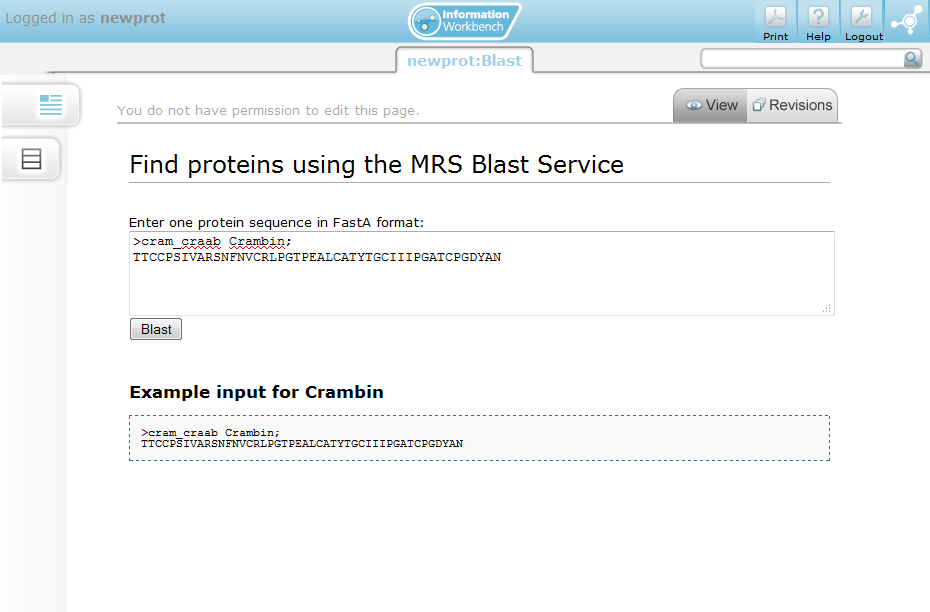
****

Figure 7: MRS Blast Interface

**I4**: Protein page

The purpose of the protein page is to visualize all available information and actions for a particular protein, including the associated PDB models and rendered images. In the SSP all proteins are uniquely identified, in particular by their Uniprot identifier in the prototypical implementation. See Figure 8 for an example protein page of Crambin.

The protein page provides the following functionality:

* Visualize available information of the protein, e.g. from Uniprot
* List associated PDB models
* Provide action components for each associated PDB model
  + Compute YASARA visualization
  + Compute hot spots (UC5)
  + Link to PDB file page
  + Compute HOPE report (UC6)
  + Compute model with YASARA (UC4)
* Import further PDB models for this protein (see Figure 9)
  + Use MRS web service to find PDB models for the current protein
  + Present results and add functionality for importing selecting model
* Protein visualization (incl. alignments) in 3DM (see Figure Figure 10)
  + Show alignments of the protein in 3DM in a new browser window
  + Commercial version only
* Visualization of Protein computed with YASARA on the right side
  + Displayed only once computed

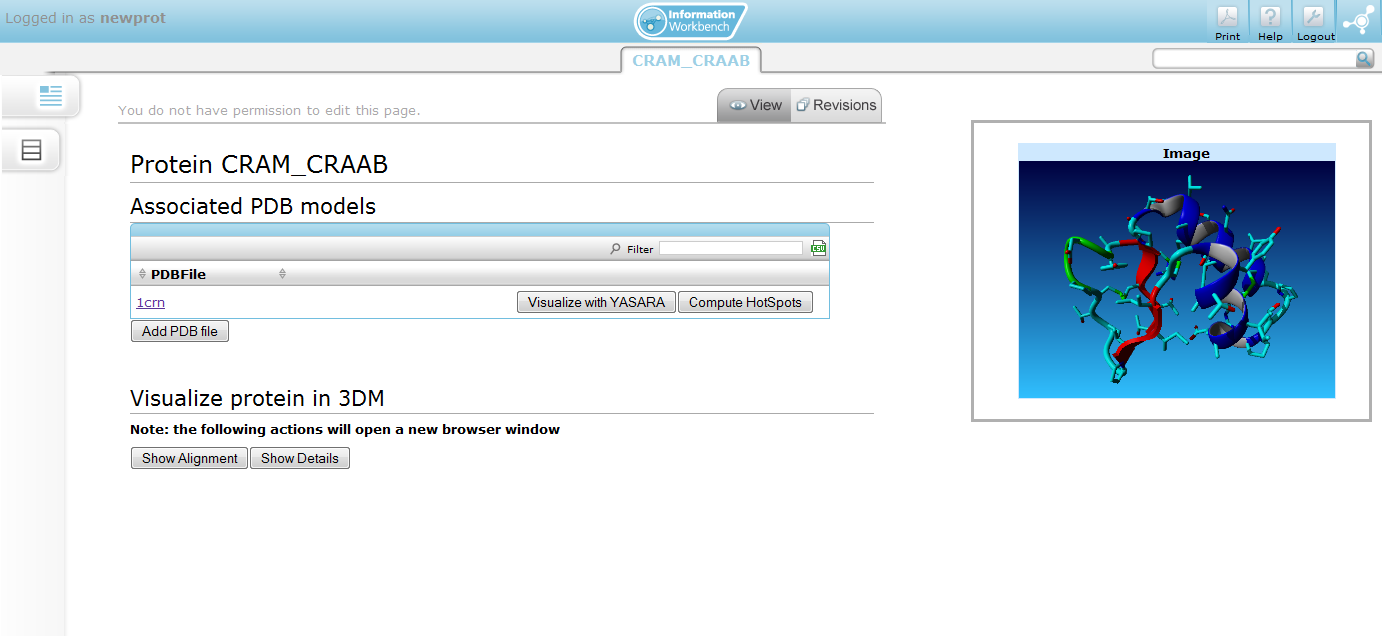


Figure 8: Protein page for protein Crambin

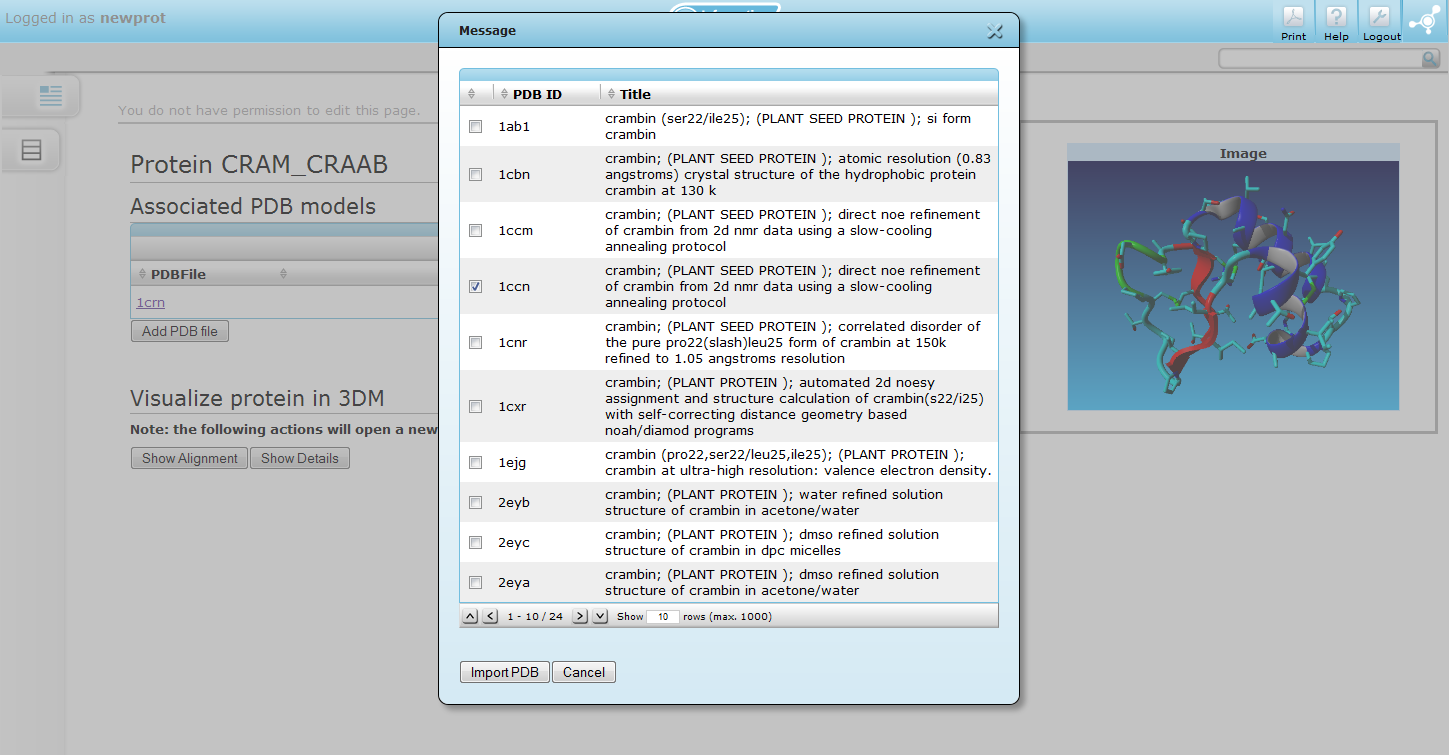


Figure 9: Import additional PDB models

****

Figure 10: Alignment visualization in 3DM

**I5**: PDB File Page

The PDB file page is one the entry point for most computational tasks, i.e. computation of hotspots and visualization with YASARA. On the PDB file page (see Figure 11 for illustration) the following actions can be performed:

* Download the PDB file
* Delete the PDB file
* Visualization of the protein with YASARA
  + Computation of a PNG and a scene using the YASARA server
  + Visualization of rendered PNG images
  + Download of YASARA scene
* Computation of hot spots and visualization of results with YASARA (UC5)
  + Trigger asynchronous hot spots computation task
  + Download hot spots results file in NPF (once the computation is finished)
  + Create YASARA visualization of computed hot spots
  + Download YASARA scene

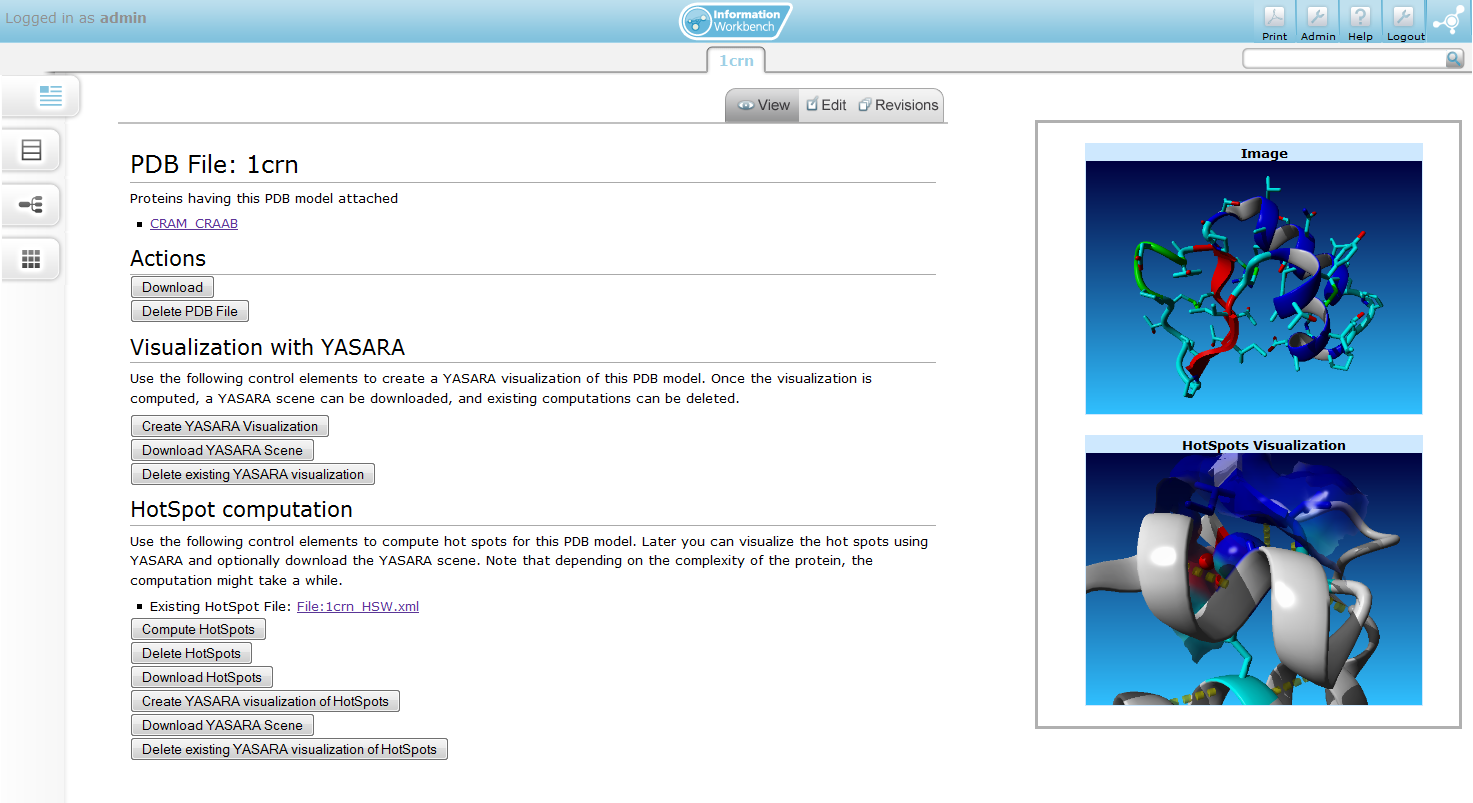


Figure 11: PDB file page

# 7. Roadmap and Next Steps

In this deliverable we have described the initial requirements and design of the SSP and provided an implementation in a first prototype.

This prototype is accessible on the web at <http://newprot.fluidops.net/>

Already the first prototype integrates the main software tools from the project partners. An important aspect for the integration of the tools was the definition of a common data format – the NewProt Format (NPF) – which provides the basis for the interoperability of the tools.

Based on the results from the validation of the portal which will be performed as part of Task 1.2 (*Validation of the interoperability of the SSP*) as well as based on user feedback, the design and implementation of the portal will be refined, and additional use cases will be identified as well as implemented. The next version of the portal will be delivered in month 24 with D1.2 (SSP Operational).

# Appendix

## I. NewProt Format Schema definition

<?xml version=*"1.0"* encoding=*"UTF-8"* standalone=*"no"*?>

<xs:schema xmlns:dc=*"http://purl.org/dc/elements/1.1/"*

xmlns:xs=*"http://www.w3.org/2001/XMLSchema"* attributeFormDefault=*"qualified"*

elementFormDefault=*"unqualified"*>

<xs:annotation>

<xs:documentation xml:lang=*"en"*>

XML Schema 2012-10-09 for

Newprot. The Newprot Format (NPF) is a XML-based file format for data

exchange between the different software tools. The format allows

representing residues as a mapping from Residue to Residue Value.

</xs:documentation>

</xs:annotation>

<xs:import namespace=*"http://purl.org/dc/elements/1.1/"*

schemaLocation=*"http://dublincore.org/schemas/xmls/qdc/2008/02/11/dc.xsd"*>

</xs:import>

<!--

ID elements for a residue

the id is constructed from components "A:32B Cys"

-->

<xs:simpleType name=*"chainidType"*>

<xs:annotation>

<xs:documentation>Molecule chain identifier, one character (case sensitive), can be empty

</xs:documentation>

</xs:annotation>

<xs:restriction base=*"xs:string"*>

<xs:pattern value=*"[A-Za-z0-9]?"*/>

</xs:restriction>

</xs:simpleType>

<xs:simpleType name=*"nameType"*>

<xs:annotation>

<xs:documentation>Residue name, one character (case sensitive, uppercase only)

</xs:documentation>

</xs:annotation>

<xs:restriction base=*"xs:string"*>

<xs:pattern value=*"[A-Z0-9' ']{3}"*/>

</xs:restriction>

</xs:simpleType>

<xs:simpleType name=*"sequenceNumberType"*>

<xs:annotation>

<xs:documentation>Residue sequence number

</xs:documentation>

</xs:annotation>

<xs:restriction base=*"xs:integer"* />

</xs:simpleType>

<xs:simpleType name=*"insertionCodeType"*>

<xs:annotation>

<xs:documentation>Residue insertion code (can be empty)

</xs:documentation>

</xs:annotation>

<xs:restriction base=*"xs:string"*>

<xs:pattern value=*"[A-Z]?"*/>

</xs:restriction>

</xs:simpleType>

<xs:simpleType name=*"pdbNumberType"*>

<xs:annotation>

<xs:documentation>PDB number

</xs:documentation>

</xs:annotation>

<xs:restriction base=*"xs:int"*>

</xs:restriction>

</xs:simpleType>

<xs:simpleType name=*"modelNumberType"*>

<xs:annotation>

<xs:documentation>Model number

</xs:documentation>

</xs:annotation>

<xs:restriction base=*"xs:int"*>

</xs:restriction>

</xs:simpleType>

<!--

Valid elements for a residue value. Extension can be created

by putting the element into the substitution group 'residueValue'

-->

<xs:element name=*"residueValue"* abstract=*"true"* />

<xs:simpleType name=*"mutType"*>

<xs:restriction base=*"xs:string"*><xs:pattern value=*"[0-9]"*/></xs:restriction>

</xs:simpleType>

<xs:element name=*"mut"* substitutionGroup=*"residueValue"*>

<xs:annotation>

<xs:documentation>Target value expressing the degree of mutability

</xs:documentation>

</xs:annotation>

<xs:complexType>

<xs:simpleContent>

<xs:extension base=*"mutType"*>

<xs:attribute name=*"reliable"* type=*"xs:boolean"* default=*"false"* />

</xs:extension>

</xs:simpleContent>

</xs:complexType>

</xs:element>

<xs:element name=*"flags"* substitutionGroup=*"residueValue"*>

<xs:annotation>

<xs:documentation>Flags for residue, i.e. [C]atalytic, [P]ocket, [T]unnel

</xs:documentation>

</xs:annotation>

<xs:complexType>

<xs:attribute name=*"catalytic"* type=*"xs:boolean"* default=*"false"* />

<xs:attribute name=*"pocket"* type=*"xs:boolean"* default=*"false"* />

<xs:attribute name=*"tunnel"* type=*"xs:boolean"* default=*"false"* />

</xs:complexType>

</xs:element>

<xs:complexType name=*"alignmentType"*>

<xs:sequence>

<xs:element name=*"type"* type=*"nameType"*></xs:element>

<xs:element name=*"count"* type=*"xs:int"*></xs:element>

</xs:sequence>

</xs:complexType>

<xs:element name=*"alignment"* substitutionGroup=*"residueValue"*>

<xs:annotation>

<xs:documentation>Alignment definition

</xs:documentation>

</xs:annotation>

<xs:complexType>

<xs:sequence>

<xs:element name=*"variant"* type=*"alignmentType"* minOccurs=*"1"* maxOccurs=*"unbounded"*></xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

<!-- Group for selecting all valid residue values -->

<xs:complexType name=*"residueValueType"*>

<xs:choice maxOccurs=*"unbounded"* minOccurs=*"1"*>

<xs:group ref=*"residueValueGroup"* />

</xs:choice>

</xs:complexType>

<xs:group name=*"residueValueGroup"*>

<xs:annotation>

<xs:documentation xml:lang=*"en"*>

This group collects valid

values for residues.

</xs:documentation>

</xs:annotation>

<xs:sequence>

<xs:choice minOccurs=*"0"* maxOccurs=*"unbounded"*>

<xs:element ref=*"residueValue"* />

</xs:choice>

</xs:sequence>

</xs:group>

<!-- Type definition of a residue -->

<xs:complexType name=*"residuetype"*>

<xs:sequence>

<xs:element name=*"number"* type=*"sequenceNumberType"* />

<xs:element name=*"chain"* type=*"chainidType"* />

<xs:element name=*"type"* type=*"nameType"* />

<xs:element name=*"pdb\_number"* type=*"pdbNumberType"* />

<xs:element name=*"insertion\_code"* type=*"insertionCodeType"* />

<xs:element name=*"model\_number"* type=*"modelNumberType"* />

</xs:sequence>

</xs:complexType>

<!-- Type definition of a value -->

<xs:complexType name=*"valuetype"*>

<xs:sequence>

<xs:choice maxOccurs=*"unbounded"* minOccurs=*"1"*>

<xs:group ref=*"residueValueGroup"* />

</xs:choice>

</xs:sequence>

</xs:complexType>

<!-- Type definition of a residueValue element -->

<xs:complexType name=*"residueValueElementType"*>

<xs:sequence>

<xs:element name=*"residue"* type=*"residuetype"* />

<xs:element name=*"value"* type=*"valuetype"* />

</xs:sequence>

<xs:attribute name=*"isHotspot"* type=*"xs:boolean"* default=*"false"* />

</xs:complexType>

<!-- head element for metadata -->

<xs:element name=*"head"*>

<xs:complexType>

<xs:choice maxOccurs=*"unbounded"* minOccurs=*"0"*>

<xs:group ref=*"dc:elementsGroup"* />

</xs:choice>

</xs:complexType>

</xs:element>

<!-- response element as child for npf -->

<xs:element name=*"response"*>

<xs:complexType>

<xs:sequence>

<xs:element name=*"residueValue"* type=*"residueValueElementType"* minOccurs=*"0"*

maxOccurs=*"unbounded"* />

</xs:sequence>

</xs:complexType>

</xs:element>

<!-- Root element for the newprot format xml document -->

<xs:element name=*"npf"*>

<xs:complexType>

<xs:sequence>

<xs:element ref=*"head"* />

<xs:element ref=*"response"* />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

1. For the Y1 prototype of the SSP we restrict to existing proteins, i.e., all proteins in the system are identified by their Uniprot URI. [↑](#footnote-ref-1)
2. http://purl.org/dc/elements/1.1/ [↑](#footnote-ref-2)
3. http://mrs.cmbi.ru.nl/mrs-5/ a [↑](#footnote-ref-3)
4. http://pdb.org [↑](#footnote-ref-4)
5. Login credentials (User/Password): newprot/newprot [↑](#footnote-ref-5)
6. http://mrs.cmbi.ru.nl/mrs-5/ [↑](#footnote-ref-6)