HORIZON 2020

INITIAL REQUIREMENTS TO REVISION CLOUD INFRASTRUCTURE

Deliverable D1.2

Circulation:	PU: Public
Lead partner:	SINTEF
Contributing partners:	DFKI, Fraunhofer, Jotne, Arctur
Authors:	Håvard Heitlo Holm, Volkan Gezer, Jochen Haenisch, Nejc Bat, Christian Altenhofen
Quality Controllers:	Daniel Weber
Version:	1.0
Date:	21.12.2015

DOCUMENT HISTORY

Version[1]	Issue Date	Stage	Content and Changes
1.0	21.12.2015	Completed	Ready to be submitted to EC.

EXECUTIVE SUMMARY

This deliverable presents the requirements for setting up a first version of the CAxMan infrastructure, based on the current version of the CloudFlow infrastructure. The current CloudFlow infrastructure is first briefly described, and then each component is mentioned in more detail together with a short description on how to make it available for CAxMan. Also described are some considerations about the interaction between the Core System Design Group, developing the infrastructure, and the other partners who develop software that will be integrated within it.

[1] Integers correspond to submitted versions

TABLE OF CONTENTS

1 INTRODUCTION

This deliverable presents the requirements for setting up a first version of the CAxMan infrastructure. The infrastructure will be based on the infrastructure developed in CloudFlow[2], as the partners developing the infrastructure are partners of both CloudFlow and CAxMan. The necessary existing foreground in CloudFlow will therefore be brought in as background in CAxMan. Components that are planned to be developed or improved in the CloudFlow infrastructure in the upcoming year will also be included in the CAxMan infrastructure as sideground as they become stable. Similarly, infrastructure features developed as foreground in CAxMan, which also improve the CloudFlow infrastructure, will be brought in as sideground in CloudFlow.

In order to provide the reader with a basic understanding of the infrastructure's role in this project, a brief and concise description of the current state of the CloudFlow infrastructure is given in Section 2.

Section 3 describes the different components of the infrastructure, based on the requirements they have to fulfil in CAxMan. The requirements span from enabling access to virtual machines and HPC resources, to deployment of infrastructure web services. More high-level requirements are also included, such as portal web pages, communication between developers, integrating future development from the CloudFlow project, etc.

The interaction with other Work Packages is described in Section 4, as most software developed in CAxMan will be executed as parts of workflows within the CAxMan infrastructure. It will be necessary to make other partners aware of their need to integrate their software within the CAxMan infrastructure at an early point, and this section describes some of the strategy towards such integration.

In Section 5 some concluding remarks are given.

User interfaces are a part of Work Package 1, but it is at this point too early in the project to describe them in detail. A proper requirement evaluation of the user interfaces will therefore be part of the deliverable D1.5 due in M18.

2 DESCRIPTION OF THE CLOUDFLOW INFRASTRUCTURE

The CloudFlow infrastructure provides a flexible platform enabling a low-effort, user-friendly integration of multi-vendor engineering services and applications in the Cloud. From the user perspective, the infrastructure is seen as a whole, but interactions between system components are rather complex. To separate its functionalities, the infrastructure is designed as a multilayer architecture with five main layers (Figure 1). Each layer has its own dedicated task and contains one or more system components. Not only hiding the complexity between end user and the infrastructure, the layer approach also allows flexible, loosely coupled and fully distributed system components. Deployment of infrastructure components is therefore completed with minimum configuration effort.

[2] Financially supported by the EU through the FP7 project CloudFlow(FP7-2013-NMP-ICT-FoF-609100)

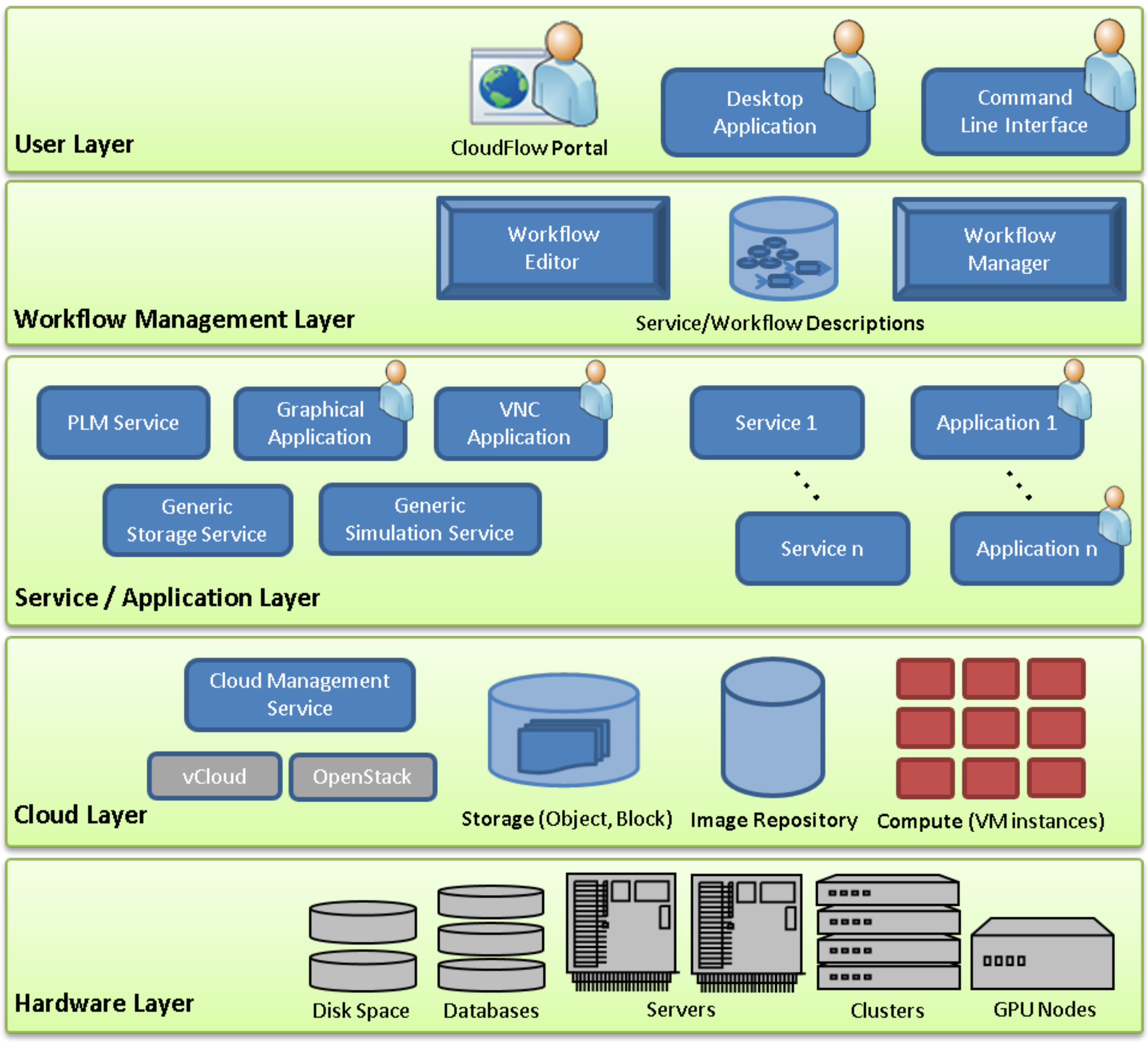

FIGURE 1 – THE LAYERED STRUCTURE OF CLOUDFLOW INFRASTRUCTURE

In CloudFlow, each experiment is described as a workflow. Workflows in CloudFlow are the chain of services that run in a sequential manner and need to be defined by using provided tools and fulfilling the requirements. They can be composed of one service or more, and even include workflows as services. Services, in this context, need to be added into the CloudFlow semantic repository using Workflow Editor which is explained in Section 3.1.1.1.

Workflows are executed, monitored, and terminated by the Workflow Manager (WFM) (Section 3.1.1.2). However, for proper handling of services as well as workflows by WFM, services are categorized into three types:

- Synchronous Services
- Asynchronous Services
- Applications

Synchronous services are short-running services and their results are returned immediately. Asynchronous services can also run for a short time, but most of the time their computation times are longer and their results are pulled periodically to check their status. Applications are asynchronous services, but they need to send their status explicitly to the WFM to inform when their execution is completed.. They also interact with the users by providing their custom GUIs defined in HTML.

Figure 2 shows a summarized interaction between the main software components of the CloudFlow infrastructure during the execution of a workflow in a high-level schematic way. When a user executes a workflow through the Portal or another client, WFM fetches the workflow contents, which are generated by the Workflow Editor during workflow design, querying the semantic service/workflow description repository. After the descriptions are fetched, WFM contacts the services locating their WSDL (Web Service Description Language) file. Independent of the service location, WFM executes services in order to complete a workflow. During workflow execution, if any feedback messages are provided by the service, they will be displayed to the user via the client.

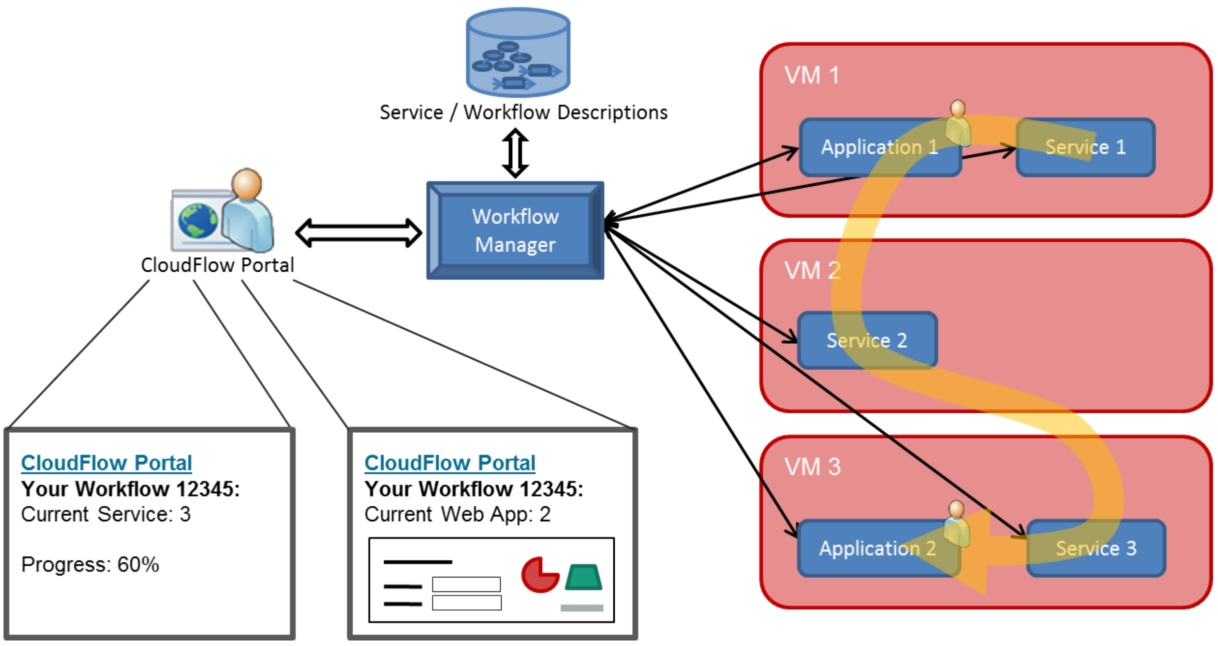

FIGURE 2 – SIMPLIFIED INTERACTION BETWEEN WORKFLOW MANAGER AND A WORKFLOW

For a more detailed description of the CloudFlow infrastructure, a CAxMan edition of CloudFlow deliverable "D800.11 V2 of CloudFlow infrastructure – Part I" has been created. This document contains detailed discussions on the adapted definitions depicted in Figure 1, as well as their current implementations. Since the D800.11 is a deliverable with restricted confidentiality level, the CAxMan edition is only made available for members of the CAxMan consortium, and can by them be found in the eRoom:

https://project.sintef.no/eRoomReq/Files/math/CAxMan/0_40fe5/D800_11_Part_I_Infrastructure_v2_CAxMan_edition.docx

3 REQUIREMENTS FOR THE INFRASTRUCTURE

3.1 INFRASTRUCTURE SERVICES

This section describes what software components the CAxMan infrastructure needs to consist of, what functionality CAxMan requires from these components and how they are related to similar components in CloudFlow.

3.1.1 Workflow Handling

Software that will be made available in the cloud needs to be deployed as web services. Each service in the infrastructure may have different purpose and may perform a special task only partially. In this case, to perform a complete task, combining services will be necessary. Combination of services requires their inputs and outputs to be compatible with each other as well as a standardized interface for a flawless execution.

To standardize the combination of services, a workflow approach is introduced. Workflows can be considered as packages, chained up with multiple services right after each other to perform specific tasks. They consist of multiple steps and at each stage a service is executed. After execution is completed, the output of the current stage is passed into the input of the next stage. When the whole workflow execution is completed, the workflow results are finally displayed to the user via the Portal or another desktop client.

The workflows are designed in such a way that they can be tracked by the infrastructure in terms of execution and permission control. To make a workflow possible to execute in the infrastructure, first the compatible services must be added and then the workflow must be designed. Both tasks can be performed using Workflow Editor (WFE) GUI or SOAP interface. Then the execution is performed using Workflow Manager (WFM).

Both tools are available in the current CloudFlow infrastructure and stable. They will also be ported to the CAxMan infrastructure in the early stages and will be enhanced when needed. Both components are written in Java and provide SOAP interfaces to allow access to their APIs.

3.1.1.1 Workflow Editor

WFE is an infrastructure component designed to perform service integration tasks and to design workflows. When accessed through the Portal (Section 3.3), its graphical user interface can be used to perform all tasks that are available as API functions.

Services to be integrated into CAxMan need to be added into the semantic repository of the infrastructure. By utilizing WFE, regardless of which machine the services are located at, their WSDL (Web Service Description Language) descriptions will be converted into machine-readable XML format and stored in the semantic repository. The data stored in the database is then used by the WFM through the life of a service/workflow and also by the WFE during a workflow design.

The current version of WFE has the following functionalities:

- Add/delete services
- Add/edit/remove workflows
- A GUI with Drag & Drop support

WFE is written as a web service, and therefore only requires a Tomcat-enabled server (minimum version 6).

3.1.1.2 Workflow Manager

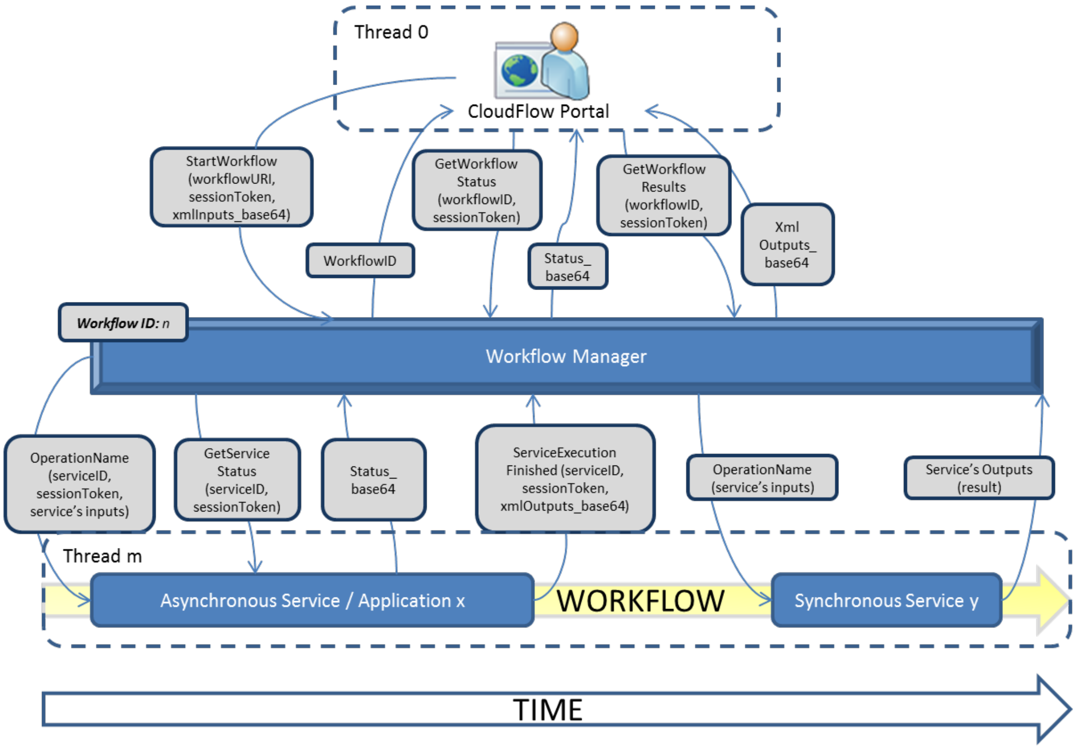
Thread 0
CloudFlow Portal
StartWorkflow (workflowURI, sessionToken, xmlInputs_base64)
WorkflowID
GetWorkflow Status (workflowID, sessionToken)
Status_ base64
GetWorkflow Results (workflowID, sessionToken)
Xml Outputs_ base64
Workflow ID: n
Workflow Manager
OperationName (serviceID, sessionToken, service's inputs)
GetService Status (serviceID, sessionToken)
Status_ base64
ServiceExecution Finished (serviceID, sessionToken, xmlOutputs_base64)
OperationName (service's inputs)
Service's Outputs (result)
Thread m
Asynchronous Service / Application x
WORKFLOW
Synchronous Service y
TIME

FIGURE 3 – THE COMMUNICATION WITH THE WORKFLOW MANAGER

The WFM plays an important role to hide the complexity of executed chain of services and applications.

The WFM starts a selected workflow (chain of services) and passes the needed input parameters through the lifecycle of a workflow. Then it monitors the workflow and asynchronous services and displays the GUI of applications to the user via the Portal or other clients, e.g. desktop client (Figure 3). After execution is completed, WFM stores final results and these can later be accessed by the user upon request.

Main features of the current version of WFM:

- Permission check on workflows/services
- Execution/termination of workflows
- Termination of asynchronous services
- Data flow between services
- Track service execution durations

WFM consists of a back-end and web service as a front-end. However, they do not need to be located in the same virtual machine.

The requirements for running the WFM back-end are:

- A Java installed virtual machine (minimum version 1.7)
- MySQL server to store service/workflow statistics
- The selected port for front-end communication must be accessible from outside

The only requirement for the WFM front-end is a Tomcat-enabled server (minimum version 6).

3.1.2 Semantic Repository

All service/workflow related information in CloudFlow is stored in a semantic database (repository) with semantic descriptions. This semantic database uses a meta-ontology defined specifically for CloudFlow (Figure 4) and based on OpenRDF-Sesame. Semantic descriptions of workflows and services improve reusability and interoperability between them and allow discovery of compatible services with minimum effort. The ontology will be adapted to CAxMan during infrastructure porting. Currently the ontology contains descriptions to allow execution of the services using semantic technologies, but throughout the project it will be extended to store semantic description of the service parameter types to increase compatibility.

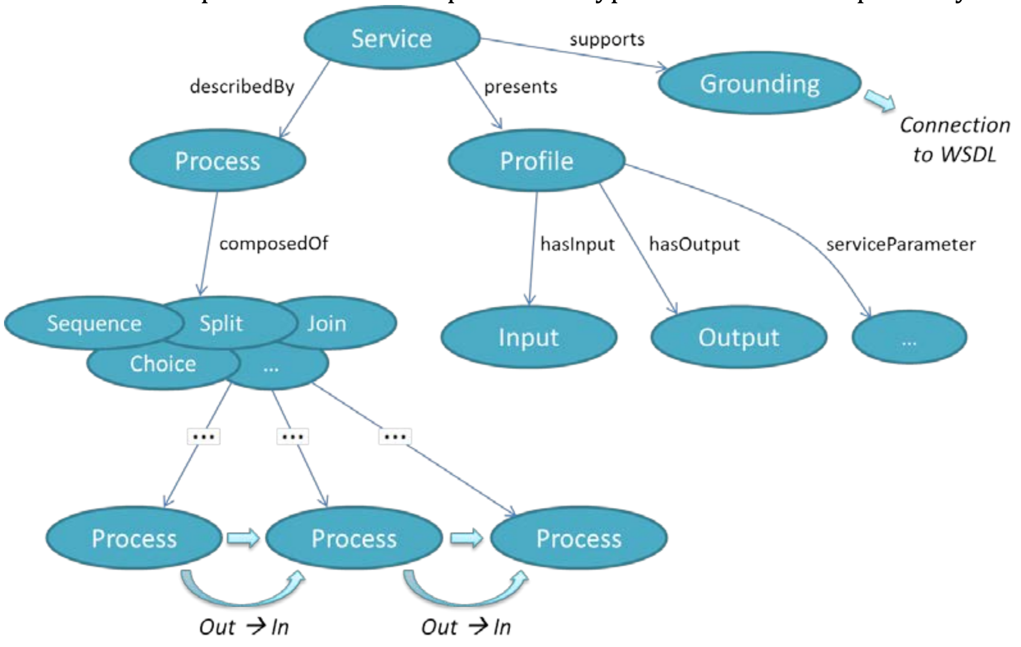

FIGURE 4 – OWL-S BASED CLOUDFLOW META-ONTOLOGY SCHEMA

The repository is deployed as a Java web service and requires only Tomcat to run. It provides its own interface and an API to communicate with.

3.1.3 Authentication

As with any Cloud based solution, the CAxMan infrastructure requires user authentication and authorization. All requests towards infrastructure components must require the client to authenticate, but in such a way that the user's username and password is protected. To fulfill this requirement OpenStack Keystone will be used, in the same way it is currently used in CloudFlow. Keystone provides token based authentication, where a user provides a username and password in return for a string that uniquely represent the user for a limited time span. This string is called the user's *token* and will be passed to all web services in the infrastructure.

Since some of the tasks that will be executed in the CAxMan infrastructure will potentially take a long time, a token with limited time span should be avoided. This requirement is also found in CloudFlow, and there is an ongoing process to redefine the usage of Keystone in CloudFlow to be able to allow arbitrary long execution times. The plan is to use tokens which can be granted restricted permissions but without a given expiration time. Such tokens could also be used for authorization.

Authentication functionality will be developed in CloudFlow by creating a set of web services that only exposes the Keystone functionality that is relevant for the project. After these web services have been implemented, they will be used in CAxMan as sideground.

3.1.4 File Handling in the Cloud

Besides Cloud computing capabilities, the CAxMan platform will also have to be able to store files and data in the Cloud. The platform will consist of several virtual machines, as well as the HPC centre and perhaps even local clients, where all of them should have access to the same files. Because of this, one or more centralized Cloud storage solutions will be used. For complex products that go through several iterations with design, simulation and redesign, metadata describing the design process should be allowed to be stored alongside the product itself. In other cases, the end user only requires simple object storage. End users will also require a simple and interactive way to organize, navigate and handle their files, and transfer them between the Cloud and their own computers.

In order to support these requirements, the PLM system described in Section 3.3 will be used alongside with OpenStack Swift object storage. However, these solutions have different APIs, while services that need to read and write files should be ignorant of which of these storage solutions they communicate with. The generic storage services (GSS) developed in CloudFlow will be used to achieve this.

GSS is a set of SOAP based web services and provides one API for performing the most common file interaction tasks across different Cloud storage providers. The main functionality of GSS is uploading and downloading of files to the Cloud. This is done by a SOAP request to GSS, followed by a REST request based on the output given from GSS. For such tasks, GSS acts like a look-up table providing structured information on how to interact directly with the requested Cloud storage. Functionality for listing files, and for creating and deleting folders, are done by GSS interacting with the storage providers directly behind the scenes, only through a SOAP request.

GSS is currently deployed in CloudFlow with back-ends for OpenStack Swift object storage and Jotne's PLM server, where both storages are hosted at Arctur. This will be the setup for CAxMan as well. A third back-end targeting WebDAV endpoints is also under development in CloudFlow, and will be enabled in CAxMan if the back-end becomes successful, and if such a back-end is needed. WebDAV exposes a RESTful API towards files stored on a regular server.

In order to deploy GSS for CAxMan only minor modifications to the source code have to be made. The modifications in the source code are typically related to changing endpoints to the CAxMan specific Swift and PLM servers. GSS will then be ready to be deployed on a virtual machine with GlassFish4.0 server, with public access through a proxy server.

3.1.4.1 File Chooser Web Application

In order to provide a natural interface towards the Cloud storage solutions, the File Chooser application developed in CloudFlow will be taken into CAxMan. It acts as a web front-end to the functionality exposed by GSS. The file chooser allows the user to navigate through folders and select files that should be used within a workflow, as well as download and upload files from and to the different Cloud storage back-ends. The application also exposes functionality for starting workflows based on a file through the file's context menu. For instance, it is possible to right click on a CAD model and start some visualization workflow.

The File Chooser consists of two parts, its web-based user interface and its web application service. The web application service wraps the web-based user interface inside the API recognised by WFM, so that the File Chooser can be used as a service and as a step inside workflows.

In the CloudFlow project, care has already been taken to ensure that the File Chooser is loosely coupled from the location of the rest of the infrastructure. The File Chooser therefore requires:

- a virtual machine with Apache2 server with PHP 5.6 with php5-json, php5-soap and php5-curl for hosting the web user interface;
- a virtual machine with GlassFish4.0 server for hosting the web application service. The same VM hosting the GSS can be used for this purpose;
- public access to the Apache server enabled through the proxy server.

When this is in place, a simple workflow has to be added to WFE in order to enable the File Chooser functionality through the CAxMan Portal.

The CAxMan project does not require any immediate changes in this service, but minor modifications are to be expected.

3.1.5 Access to HPC Resources

When executing computationally intensive simulations, such as thermal and stress analysis in WP3, HPC resources are required in order to have good performance across several computing nodes. These resources will have to be utilized seamlessly as a step within a complex workflow of different tasks, and a WFM compatible web service interface for executing HPC jobs is therefore required. This must also include a mapping between a CAxMan user and a user in Arctur's HPC centre, which should be hidden from the end user.

A generic asynchronous service has been developed for this purpose in the CloudFlow project, which is currently deployed in the form of a first working version. The service provides an interface where the most important details of the job submission script, such as number of cores and nodes, maximal duration and command lines to be executed by the HPC nodes are taken as input. The service will then proceed by inserting this information into a job script, mapping the CloudFlow user to an HPC user, and submitting the job to the HPC queue. In order to allow the asynchronous service to report the status and the result, the submitted job is required to report its status according to the HPC service's documented API.

As the input and output parameters to the HPC service are different from the natural input and output parameters of the application that is executed, application specific pre- and post-processing services should be added. These will be in charge of massaging the natural input parameters into those required by the HPC service, and similarly extract output parameters. A workflow can then be created from the three services (pre-processing, HPC service and post-processing) where the workflow has the natural input and output parameters with respect to the underlying application. This workflow then represents a single building block that can easily be used as a step in other workflows.

The HPC service will be adapted from CloudFlow and used in CAxMan as well. Since the current state of the implementation is a first working version, it should be expected that both the implementation details and the service's API might be required to change. A GlassFish4.0 server is required for deploying the web services, and a MySQL database is required in order to keep track of the submitted tasks.

3.1.6 File Format Conversion Services

Within the CAxMan project, data will be produced and stored in many different representations, such as tri-variate NURBS, B-Reps, subdivision volumes, tetrahedral/hexahedral meshes or voxel representations, each having its own file format. Depending on the tool used to perform a specific action, there can also be multiple different file formats for the same representation, e.g. different formats for CAD models when working with different CAD systems (TopSolid, Catia, AutoCAD, and SolidWorks).

The use cases within the CAxMan project and the envisaged workflows often require the subsequent execution of applications where the output format of one application is not supported as an input format of the next one. Therefore, the infrastructure should provide a set of conversion services that can be called during the workflow between two applications to overcome incompatibilities of formats and representations. These services could either be converters that transform the same representation from one file format to another (as for the example with different CAD formats mentioned above), or they could even convert the data between representations, usually also resulting in a different file format. One such service already exists in CloudFlow, converting files from the STEP format to G2. However, in order to use it, the service currently has to be manually added to workflows.

Based on a semantic description of the supported input and output formats of all applications offered in the Cloud infrastructure, as well as of all available conversion services, the Workflow Manager could automatically choose the correct converter service for a given workflow and insert it as an additional step where needed.

The semantic modelling is already planned within the CloudFlow project and could be used for CAxMan. The envisaged set of conversion services will be introduced in CAxMan but could also be offered as CloudFlow services afterwards.

3.1.7 VNC Connection

In some cases, the functionality of a desktop application is too complex to be made available as a web interface, and in those cases there are two options for how the application can be integrated in the CAxMan infrastructure. The first option is that only a subset of the functionality can be made available through web services, providing functionality only for dedicated tasks to be performed as part of a workflow. The second option is to create a VNC session through a WFM compatible web service, allowing the original desktop application to be accessed within a workflow.

A VNC service has been integrated in CloudFlow, and can be used in CAxMan as well, but the service should be revisited in order to become more generic. Currently, the existing VNC service does not support any input or output parameters, and the VNC session is not able to interact with WMF. The integration with the infrastructure is therefore quite weak. The revisited version should be able to handle input and output parameters in a similar way as the HPC service, and experience from this job should be used to create a generic solution also for VNC.

It is at this point not clear whether VNC services will be required by CAxMan or not. The implementation of a revisited VNC service will therefore be made in CAxMan if needed, unless this job has not been already done in CloudFlow at that point.

3.1.8 Registration of Resource Utilization and Billing

In order to develop a commercial product out of CAxMan, all executions of workflows and services, as well as usage of HPC resources and storage, must be registered. Customers of CAxMan should then receive bills based on this resource usage according to the business model developed as part of this project.

The CloudFlow project has had the same requirements, and there is ongoing activity on implementing such functionality. This functionality will be provided in two different sets of services, an accounting part for monitoring resource usage, and a billing part for translating the monitored usage into euros according to the business model. As these services are not yet implemented, the decision whether they can be used as is, or if they have to be modified, has to be made at a later stage. It is also expected that the business models for CloudFlow and CAxMan will differ to an extent where the billing services might have to be re-implemented from scratch. The accounting services will however be independent from the business models, and the functionality for accounting is therefore believed to be reusable by CAxMan.

3.2 CAXMAN PORTAL

In order to provide a public endpoint where users can interact with the CAxMan infrastructure, a web portal is required.

The CAxMan Portal is the web site where users log in, interacts with their workflows, and design new workflows. It therefore acts as a client to the WFM and WFE (see Section 3.1.1), and makes sure that the user is logged in to the authentication services (Keystone). The user should also be able to interact with some account settings and billing through the Portal.

The current state of CloudFlow is that users and project partners interact with a version of the Portal that has been developed mainly to support the functionality needed for developing, testing, and executing workflows and new infrastructure components through the web browser. This is a stable version, even though it is not designed to scale to public usage. In order to handle user requirements better instead of only developers' requirements, a new version of the Portal is currently being developed for CloudFlow.

In order to get a working portal up and running for the CAxMan project as fast as possible, the current CloudFlow Portal will be ported to the CAxMan infrastructure. The visual impression will be slightly changed with respect to the colours found in the CAxMan logo and on the CAxMan website before it is deployed inside the CAxMan infrastructure. Two versions of the Portal will be available, where the first represents a stable version for production usage, and the second is meant for development and testing of new infrastructure features.

After the new version of the Portal has been made available for CloudFlow and has been through a few iterations of bug fixing, the new version will be modified and made available for CAxMan. This transition will require that the Keystone services are ready for both CloudFlow and CAxMan.

3.3 PLM

The PLM solution in CAxMan will be a development based on the application used in CloudFlow, that is, the Jotne commercial product EDMopenSimDM. EDMopenSimDM is designed to manage PLM/CAD/CAE information for engineers. It is a master data unified repository in which product and process information from many sources (such as systems, companies, etc.) can be collected and managed. The EDMopenSimDM repository is designed to handle many product versions and configurations and to distinguish between information packages received from multiple suppliers and partners delivered to many customers. EDMopenSimDM applies ISO 10303 for long term archiving and retrieval and for services that interoperate not only with users, but also with other tools, for example in a Cloud or Big Data environment.

The developments in this project will make the PLM application the central data management solution for:

- multi-disciplinary design and optimization during the product development process;
- life-cycle configuration of product data by integration of multi-disciplinary engineering and manufacturing data from different sources into one data set;
- partly automated engineering workflow through ontology based service-oriented architecture (tool-to-tool communication);
- supply chain integration through collaboration solutions;
- knowledge management beyond the duration and beyond the scope of single product development processes.

The PLM application consists of a server part and two alternative user interface parts.

3.3.1 PLM Server

The PLM server includes a configured version of the Jotne product EDMserver and the optional 3D visualization tool VCollab. The PLM server runs on MS Windows computers with the following recommended configuration:

- Hardware Requirements:

  32 GB RAM;
  500 GB free disk space on the computer hard drive. This includes the software and needed temporary space for running the server, but this does not include space for back-ups.
  These hardware recommendations are based on a system that runs one EDMserver™ with three EDMapplicationServers™. The parameter that affects performance the most is the size of the available virtual memory and disk access speed. Another factor that affects performance is the typical size of CAD/CAE/CAM STEP files that are imported, converted and validated.

- Software Requirements:

  64-bits Operating System:
  Windows 7, Windows 10
  Runtime environment:
  Java Runtime Environment 1.8 (JRE 8) 64-bit

The PLM server offers all its functionality as SOAP web services, using HTTP and TCP as communication protocols and XML as data format of the delivered data. In addition, some file handling services are implemented as REST web services; REST is based on a stateless communication protocol like HTTP and allows only the fundamental HTTP messages. Web service functionality will be used for tool-to-tool communication in the Cloud.

3.3.2 PLM Client

The PLM solution comes with two end user clients and graphical user interfaces: one rich client that offers all EDMopenSimDM functionality to the engineer, and one web-client with a subset of functions, which is typical needed for supplier/customer collaboration.

The following configuration is recommended for the Windows client:

- Hardware Requirements:
  8 GB RAM with 12 GB Virtual Memory;
  A graphic accelerator card with minimum of 256 MB RAM (for the optional VCollab);
  Graphic drivers that support OpenGL 1.2 and above.

- Software Requirements:
  64-bits Operating System:
    Windows 7, Windows 10
  Runtime environment:
    Windows .NET Framework 3.5 SP1

3.3.3 Possible Extension of PLM Access through GSS

In CloudFlow, files may be handled in the PLM application by three ways:

- through GSS (upload/download/file listing etc.),
- through the PLM server specific web-services (all PLM functionality through SOAP and some through REST) and
- through the PLM web-client.

Thus, services implemented with GSS can access files on either the OpenStack Swift or on the PLM servers. However, this current integration with GSS is limited to file management and does not exploit the meta-data environment that makes PLM especially valuable.

Based on the requirements from the other Work Packages, the possibilities of enabling versioning, folder properties and other meta-data interaction on the PLM server also through GSS will be looked into. At the time of writing, it is still not clear whether such extensions will be necessary, but the possibility to include new developments is kept open.

3.4 CLOUD AND HARDWARE COMPONENTS

Arctur will act as the main infrastructure provider that will enable the appropriate computing environment for the execution of the project within the currently available computing means and the additional dedicated hardware.

The infrastructure consists of three main components of the system:

1. The HPC physical infrastructure with the following characteristics:
Log-in node: intended for logging on the system, launching jobs, compiling codes and modules, testing.
Standard node:

- multiple IBM iDataPlex dx360 DualCPU servers
- Intel Xenon X5650 processors (6 cores @ 2.66 GHz, hyperthreading OFF)
- 32 GB memory per server (2.67 GB per core)
- InfiniBand QDR 40 Gbps interconnections
- Disk array comprised of a \home partition that is backed up and an expandable \scratch partition that is not backed-up.

»Fat« node: same configuration as standard nodes but with a memory increase to 144 GB for accommodating specific memory intensive tasks.

2. Two physical GPGPU nodes with the following characteristics:
GPU0: NVIDIA-GPU server:
- 2x Opteron 6272 (2x 16 cores)
- 192 GB RAM @ 1600 MHz
- NVIDIA Quadro 4000 GPU
- NVIDIA Tesla C2050 GPU
- NVIDIA Tesla K40 GPU

GPU1: AMD-GPU server:
- 2x Intel x5650 (2x 6 cores)
- 192 GB RAM @ 1333 MHz
- 2x AMD FirePro S9150 GPU

3. Virtual Cloud computing infrastructure composed of both VMware and OpenStack infrastructure utilising the appropriate components of individual systems.

This system is the foundation for all CAxMan infrastructure and its components. The virtual Cloud computing infrastructure hosts the Portal, the Workflow Manager components and the underlying services that this system spawns and controls. The services that require physical machines run on either the HPC system or on the GPGPU servers.

To guarantee the best possible level of security and availability of the system, solutions like e.g. proxy forwarding and firewall systems are set up especially for the CAxMan environment, in addition to the standard firewall and certificate system. Therefore, the system is protected against intrusions at the maximum possible level, given the available resources.

To guarantee the uninterrupted operation of the developed components, the environment is split up into two distinct parts: one for testing and one for production. Only after a component is tested it is migrated and put into production. Relevant Virtual Machines are also regularly backed up with a predefined number of backups saved.

To enable the appropriate Cloud Elasticity, the suitable Cloud management components such as load balancers, fast provisioning and automatic deployment components will be used to guarantee a dynamic and scalable operation of the system. The OCCI (http://occi-wg.org/) guidelines are observed for this intent.

The OCCI guidelines are already implemented in the individual components. The guidelines regarding system actions and linking of individual components are implemented in the Workflow Manager (Section 3.1.1.2) which provides the correct workflows for execution. The guidelines regarding networking and network configuration are handled by the VMware and OpenStack systems respectively.

Furthermore, some components that are being developed within the project extend the existing OCCI guidelines. One such component is the accounting component (Section 3.1.8) that will capture, gather and store the resource usage for further charging the end user of the system.

3.5 COMMUNICATION

In order to get a consistent and well-organized development of the CAxMan infrastructure, a **core System Design Group (cSDG)** based on the 5 partners involved in WP1 (SINTEF, Fraunhofer, DFKI, Arctur and Jotne) is defined. In order to have continuous development the group will have bi-weekly teleconferences led by DFKI, who is the Work Package leader.

A good, structured and easy to use system is required for following up bugs and issues related to the infrastructure to ensure that problems are properly dealt with. A private GitHub repository is created and will be used for this purpose inside the CAxMan organization on GitHub[3]. Development issues will here be marked with labels showing what infrastructure components are related to the issue, as well as deadlines for resolving the issues represented by GitHub's milestones. Each issue will have a comment section, and the progress of each issue will be tracked continuously as well as in the biweekly teleconferences in this comment section.

The CAxMan organization on GitHub will also be used for sharing and distributing template and skeleton code projects for the other partners to use. These will typically represent the different service types recognized by WFM, and could also be used as tutorials for training purposes for new partners and/or future customers.

Since CAxMan will take advantage of the CloudFlow project, it is important that development of new functionality in CloudFlow is smoothly integrated in CAxMan, also after the first version of the CAxMan infrastructure is deployed. This should be fairly achievable since all 5 partners in the CAxMan cSDG are central partners of CloudFlow as well. Simultaneously, new infrastructure functionality developed in CAxMan and that would fit well into CloudFlow, should also be integrated into the CloudFlow infrastructure.

When the CAxMan infrastructure is deployed and the other partners start to expose their work as services for the WFM, these partners should also be included in the bi-weekly teleconferences. The meetings should then start with a discussion on the integration of software into the infrastructure, and then end with a discussion around the development of the CAxMan infrastructure with only the cSDG present (modelled from the technical teleconferences in CloudFlow).

4 INTERACTION WITH OTHER WORK PACKAGES

A lot of software will be developed in the other Work Packages of CAxMan, and most of this software is expected to be integrated into the CAxMan infrastructure as outlined in the previous sections. It will therefore be important that the partners providing software are aware early on that their software will be integrated into this infrastructure, and what requirements this sets to the software.

This requires that all partners are active in their integration towards the infrastructure, but it also requires that the core System Design Group provides documentation, support and training material for such tasks.

[33] http://github.com/CAxMan

4.1 WEB SERVICES

The most important requirement is that all software will have to be adapted as web services with APIs recognised by WFM, as

- synchronous services,
- asynchronous services or
- applications,

as described in Section 2.

For the long running computations, which will be asynchronous services, a proper way to report the progress will have to be considered. The status should ideally be provided as user friendly application specific information about the progress of the computational task in HTML. Computations that will be executed in the HPC centre will have to produce both final output and intermediate status report according to the API of the HPC services. This API can be revisited if it is seen necessary, but such revision should still be generic and should be done within the boundaries set by asynchronous services and the HPC queuing system.

The workflow philosophy should also be used early on, meaning that separate steps during execution of the software should be identified and regarded as individual steps in a workflow. If each of these steps can be reused in other workflows as well, these steps should be exposed as independent web services, and a workflow should be created in WFE for executing the entire task.

For the software adaptation to web services, several requirements must be considered:

- Certain software functionality must be encapsulated in independent services. These services must depend on the declared inputs.
- All types of services must be realized as SOAP web services and must provide an accessible WSDL file.
- The services must be deployed into a Cloud virtual machine that must be accessible from WFM.

For a full list of requirements, CAxMan partners should refer to the CloudFlow deliverable "D800.11 V2 of CloudFlow Infrastructure – Part I – CAxMan edition", which for the CAxMan partners can be found at

https://project.sintef.no/eRoomReq/Files/math/CAxMan/0_40fe5/D800_11_Part_I_Infrastructure_v2_CAxMan_edition.docx

4.2 INTERACTION WITH THE CLOUD STORAGE

When software is wrapped into web services and moved to the Cloud, file handling has to be considered a bit differently compared to traditional execution on work stations. Data will be stored in dedicated Cloud storage solutions (such as PLM and OpenStack Swift), and will need to be transferred between the storage server and the virtual machine/HPC node running the software. In order to get input and output data transfer correctly integrated into the software, the use of GSS is encouraged, and there are clients for GSS available in Java, C++ and Python. Using GSS typically requires the software to take file identifiers used in GSS as input instead of file names.

4.3 RESOURCES

Since the beginning of the CloudFlow project, documentation and training material have been collected and distributed based on requirements and questions from CloudFlow project partners. As the same infrastructure will be used for CAxMan, the same, or slightly modified, training material will be made available to the CAxMan partners.

The following folder in the eRoom will be used for distribution of material on the infrastructure as the different components are made available in the CAxMan infrastructure:

https://project.sintef.no/eRoom/math/CAxMan/0_3f800

In order to contribute in the process of creating WFM compatible web services, skeleton code will be made available on the project's GitHub organization, at http://github.com/CAxMan/.

5 CONCLUSIONS

This deliverable has described the infrastructure requirements for the CAxMan project. As the project by design will build on top of the infrastructure developed for the CloudFlow project, several software components from CloudFlow already fulfil the requirements for the CAxMan infrastructure. Some CloudFlow components that will be used in CAxMan, either as is or with only minor modifications, are

- Workflow Editor and Workflow Manager for workflow orchestration and execution,
- the Generic Storage Services for file handling in the Cloud,
- the HPC service for enabling access to the HPC resources within workflows, and
- services and applications for accessing PLM through the infrastructure.

Additionally, services for authentication and resource registration will be adapted in the CAxMan infrastructure as sideground as these services are developed in CloudFlow. The end user interface towards the CAxMan infrastructure will be through the CAxMan Portal, which also will be based on the Portal developed for CloudFlow. A set of services for converting file formats or data representations is identified as a requirement that will not be fulfilled by any existing CloudFlow components, and such services will be implemented as part of the CAxMan project.

One of the most challenging tasks related to the infrastructure will be to keep infrastructure components that are used in both CAxMan and CloudFlow synchronized. When a common component is improved in one of the projects, the other project should benefit from this improvement as well. A good strategy for avoiding duplication of code and divergent development will therefore be necessary.