Large-scale EXecution for Industry & Society

Deliverable D2.3

Report of LEXIS Technology Deployment – Intermediate co-design

Co-funded by the Horizon 2020 Framework Programme of the European Union
Grant Agreement Number 825532
ICT-11-2018-2019 (IA - Innovation Action)

DELIVERABLE ID | TITLE	D2.3 | Report of LEXIS Technology Deployment – Intermediate co-design
RESPONSIBLE AUTHOR	Yuanyuan Li (LINKS)
WORKPACKAGE ID | TITLE	WP2 | LEXIS Requirement Definition and Architecture Design
WORKPACKAGE LEADER	Bull/Atos
DATE OF DELIVERY (CONTRACTUAL)	31/03/2020 (M15)
DATE OF DELIVERY (SUBMITTED)	14/04/2020 (M16)
VERSION | STATUS	V1.3 | Final
TYPE OF DELIVERABLE	R (Report)
DISSEMINATION LEVEL	PU (Public)
AUTHORS (PARTNER)	LINKS, CYC, Bull/Atos, O24, LRZ, IT4I, ECMWF
INTERNAL REVIEW	Sean Murphy (CYC); Cédric Koch-Hofer (Bull/Atos)

Project Coordinator: Dr. Jan Martinovič – IT4Innovations, VSB – Technical University of Ostrava
E-mail: jan.martinovic@vsb.cz, **Phone:** +420 597 329 598, **Web:** https://lexis-project.eu

DOCUMENT VERSION

VERSION	MODIFICATION(S)	DATE	AUTHOR(S)
0.1	Table of Contents and filled-in draft (first proposal) created	04/02/2020	Yuanyuan Li (LINKS)
0.2	Filled Section 2.2.1 on LEXIS AAI	18/02/2020	Frédéric Donnat (O24)
0.3	Filled in various LRZ contributions asked for; reformulated LRZ comment to DDI-AAI lesson learned.	20/02/2020	Stephan Hachinger (LRZ)
0.4	Added introduction and summarisation on all sections	20/02/2020	Yuanyuan Li (LINKS)
0.5	Added the state-of-affairs about FPG in Section 2.3.3	21/02/2020	Huy-Nam NGuyen, Marc Levrier (Bull/Atos)
0.6	Reviewed the whole document. Fixed typos and added cosmetic for clarity in the whole document.	24/02/2020	Marc Levrier (Bull/Atos)
0.7	Accepted the modification from Marc Levrier (Bull/Atos) in the whole document	24/02/2020	Yuanyuan Li (LINKS)
0.8	Added Section 2.3.2 on Burst Buffer	25/02/2020	Cédric Koch-Hoffer (Bull/Atos)
0.9	WP7 LL on NWP for extreme events impact assessment and mitigation	27/02/2020	Emanuele Danovaro (ECMWF)
0.91	Entirely rewrote two LL in Sections 3.3.1 and 3.3.2 to match Jan's requests	28/02/2020	Marc Levrier (Bull/Atos)
0.92	Revised LL on orchestrator	04/03/2020	Alberto Scionti (LINKS)
0.93	Based on the comments from Sean and Cédric: (i) Updated the sections on DDI (ii) Updated the sections on Orchestrator (iii) Updated the section on Data Storage (iiii) Polished the entire document	12/03/2020	(i) Stephan Hachinger (LRZ) (ii) Alberto Scionti (LINKS) (iii) Martin Golasowski (IT4I) (iiii) Yuanyuan Li (LINKS)
0.94	Fixed the FPGA section and 2 LLs	16/03/2020	Marc Levrier (Bull/Atos)
0.95	Revised the section on AAI and checked the entire document	18/03/2020	Yuanyuan Li (LINKS)
0.96	Revised and fixed section 2	18/03/2020	Marc Levrieri (Bull/Atos)
1.0	Final version, all comments and requests from reviewers resolved.	26/03/2020	Yuanyuan Li (LINKS)

1.1	Added a footnote stating the difference between LEXIS Portal Front-End and LEXIS Front-End	31/03/2020	Sean Murphy (CYC)
1.2	Final check of the document	07/04/2020	Kateřina Slaninová (IT4I) Jan Martinovič (IT4I)
1.3	Revise according to the comments	09/04/2020	Stephan Hachinger (LRZ) Jan Martinovič (IT4I) Yuanyuan Li (LINKS)

GLOSSARY

ACRONYM	DESCRIPTION
AAI	Authentication & Authorization Infrastructure
A4C/ALIEN4CLOUD	Application LIfecycle ENablement for Cloud (https://alien4cloud.org)
CDI	Collaborative Data Infrastructure
CEPH	Scalable distributed storage system (https://github.com/ceph/cep)
CF	Climate and Weather Forecast
DDI	Distributed Data Infrastructure
DDN	Data Direct Networks
DSS	Data Science Storage (DSS)
ECMWF	European Centre for Medium-range Weather Forecast
EUDAT	European Data Infrastructure
GRIB	General Regularly-distributed Information in Binary form
HEAPPE	High-End Application Execution
HPC	High Performance Computing
IAAS	Infrastructure as a Service
IAM	Identity and Access Management
IME	Infinite Memory Engine
IT4I	IT for Innovation (Ostrava, Czech Republic)
LL	Lesson(s) Learned
LRZ	Leibniz Supercomputing Center (in Garching, Munich area, Germany)
NFS	Network File System
NVME	NVM (non-volatile Memory) Express, usually used as interface to SSDs
NVMEOF	NVMe-Over-Fabrics
NWP	Numerical Weather Prediction

NVRAM	Non Volatile Random Access Memory
OIDC	OpenID Connect protocol
POSIX	The Portable Operating System Interface
RBAC	Role-based Access Control
RDMA	Remote Direct Memory Access
REALM	A realm manages a set of users, credentials, roles, and groups.
ROCE	RDMA over Converged Ethernet
SAML	Security Assertion Markup Language
SBF	Smart Bunch of Flash
SME	Small & Medium Enterprises
TOSCA	Topology and Orchestration Specification for Cloud Applications (by OASIS, http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html)
TSUNAWI	Tsunami software simulator; origin, propagation and physical impacts
WCDA	Weather and Climate Data API
WP	Work Package
YORC	Ystia Orchestrator (https://github.com/ystia/yorc)

TABLE OF PARTNERS

ACRONYM	PARTNER
Avio Aero	GE AVIO SRL
AWI	ALFRED WEGENER INSTITUT HELMHOLTZ ZENTRUM FUR POLAR UND MEERESFORSCHUNG
BLABS	BAYNCORE LABS LIMITED
Bull/Atos	BULL SAS
CEA	COMMISSARIAT A L ENERGIE ATOMIQUE ET AUX ENERGIES ALTERNATIVES
CIMA	Centro Internazionale in Monitoraggio Ambientale - Fondazione CIMA
CYC	CYCLOPS LABS GMBH
ECMWF	EUROPEAN CENTRE FOR MEDIUM-RANGE WEATHER FORECASTS
GFZ	HELMHOLTZ ZENTRUM POTSDAM DEUTSCHESGEOFORSCHUNGSZENTRUM GFZ
IT4I	VYSOKA SKOLA BANSKA - TECHNICKA UNIVERZITA OSTRAVA / IT4Innovations National Supercomputing Centre
ITHACA	ASSOCIAZIONE ITHACA
LINKS	FONDAZIONE LINKS / ISTITUTO SUPERIORE MARIO BOELLA ISMB
LRZ	BAYERISCHE AKADEMIE DER WISSENSCHAFTEN / Leibniz Rechenzentrum der BAdW
NUM	NUMTECH
O24	OUTPOST 24 FRANCE
TESEO	TESEO SPA TECNOLOGIE E SISTEMI ELETTRONICI ED OTTICI

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

EXECUTIVE SUMMARY

The LEXIS project is developing solutions to integrate hardware and software components from HPC, Cloud and Big Data domains into powerful computing and Data Management platforms, including validation within real-life scenarios (Aeronautics, Tsunami Prediction, Weather Forecast) and providing easy access to Industries, SMEs and Academia through LEXIS Portal.

By month 15, the LEXIS project has deployed a set of technical solutions. The evaluation of existing solutions together with existing technical constraints has been done in Deliverables D2.1 [1] and D2.2 [2]. However, various issues and challenges were encountered during the implementation and integration processes. Lessons learned (LL) keep track of challenges and the difference between expected results and actual achieved ones, as well as the solutions and corrected actions that were / to be carried out in case of gaps.

Position of the deliverable in the whole project context

This deliverable is part of Task 2.2 - Co-design and Lesson Learned. It summarises the key components of the LEXIS architecture and the lessons learned which were identified during the co-design processes, providing initial feedback to the technical selections and integrations.

The main contributors of this deliverable are the “technical work packages”, namely WP2, WP3, WP4 and WP8:

- LINKS as the coordinator of co-design tasks and work package leader (WP4),
- Bull/Atos as the work package leader as well as for Burst-Buffers, Orchestration and FPGA related topics (WP2),
- IT4I as one federated HPC service provider and project lead,
- LRZ as the other federated HPC service provider and work package leader (WP3),
- O24 for security related aspects and CYC for the portal related topics and work package leader (WP8).

Besides, LINKS, Bull/Atos, LRZ, CYC, and ECMWF contributed to the section regarding Lessons Learned.

Description of the deliverable

D2.3 first provides a brief overview of the LEXIS architecture as it is in M15, focusing on the key components which implement new solutions or represent non-trivial issues (Section 2). Then the lessons learned during the deployment of the key components are described (Section 3).

The deliverable closes with a short summary on the technology achievements that have been made and an outlook on the development progress (Section 4).

1 INTRODUCTION

About the present deliverable (D2.3)

WP2 - *LEXIS Requirements Definition and Architecture Design* evaluates the technical requirements of large-scale pilots and the existing hardware useful for the LEXIS project with the aim to enhance the technologies to allow easy use and running HPC/Cloud/Big Data applications. Two submitted deliverables D2.1 [1] and D2.2 [2] from WP2 - Task 2.1 *Infrastructure Evaluation and Key Technology Evaluation* have reported the key existing and new parts of LEXIS platform. Another task: Task 2.2 *Co-design and Lesson Learned* concerns the co-design process, which involves many iterations with other tasks and work packages. As an intermediate report on the task in M15, this deliverable reports the key components of the finalized LEXIS architecture and includes Lessons Learned (LL) from the co-design process during the first half of the project.

The co-design process attempted to actively involve all stakeholders in the design process to help ensure the results meet their needs and are usable. To describe the design choice, the LEXIS platform provides a three-layer architecture showing how hardware and software technologies interact with each other. For getting a brief overview of our technology solutions, the report will display the architecture diagram composed of the key components and their interactions, and then introduce the deployment status of the key components in each layer. Since the LEXIS platform architecture is joined by various technical components, a set of trade-offs must be taken to guarantee the compatibility and the overall good performance. Besides, from the deeper understanding of requirements and capabilities of underlying technologies to the actual deployment, various challenges were encountered, and the corresponding solutions were proposed and are in the implementation processes. The lessons learned recorded some of these best practices, differences between foreseen objectives and achieved ones, and challenges, which serve for improving implementation of newly proposed LEXIS approaches, preventing failures and keeping the LEXIS development on schedule.

Structure of the document

The document is structured as follows:

- The first part of the deliverable (**Section 2**) will highlight the key components of the LEXIS platform describing the development status,
- The second part (**Section 3**) will depict the best practices, challenges and the corresponding solutions with respect to the implementation and integration of the key components through LL,
- Finally, a summary will be provided in **Section 4**.

2 LEXIS – PLATFORM ARCHITECTURE

LEXIS is building a platform at the confluence of HPC, Cloud and Big Data which leverages large-scale geographically distributed resources from existing HPC infrastructure, employs Big Data analytics solutions and augment them with Cloud services. The LEXIS platform lays on a three-layer architecture. A macroscopic description of this architecture is proposed in Figure 1.

These three layers are:

- The LEXIS Portal Layer, providing easy access to the LEXIS platform for Pilots and possible external users,
- The LEXIS Services Layer, running on top of the infrastructure layer. It includes federated security Infrastructure (Authentication & Authorization Infrastructure, AAI), data management (Data Distribution Infrastructure, DDI), and orchestration services (Orchestrator),
- The HPC/Cloud Infrastructure Layer, focusing on the interactions among HPC and Cloud hardware systems to provide the computing power and data storage space to the upper layers. It is implemented as a federation of multiple HPC providers and data centres.

The LEXIS Front-End[1] provides the user interface of the LEXIS platform while the LEXIS Back-End provides the content to the Front-End system part through the API. This gives the architecture a low binding between Front-End and Back-End parts of the system, which will ease the deployment of a new version of Front-End client (Web, Mobile platform) in the future.

As the LEXIS central storage library, DDI provides data virtualisation and discovery and workflow automation secured by the AAI. The connection between the LEXIS Front-End and the DDI is necessary to support listing of directories on DDI and download files to and from DDI. The connection is not established directly but through a LEXIS API Proxy (with load balancing and web filtering capabilities). The orchestrator is responsible for managing LEXIS workflows. The Front-End and orchestrator are also connected through LEXIS API Proxy to define LEXIS workflows, to deploy and execute them. To provide accounting and billing service, the LEXIS Portal is connected with the service provided by CYC.

The orchestrator needs to dynamically select available and suitable computing resources for LEXIS workflow execution. The specific HPC centre provides computing resources, which can be nodes from HPC clusters or from parts of the cloud (virtual) infrastructure. The orchestrator uses the HEAppE middleware for execution and managing LEXIS workflows parts that must run on an HPC infrastructure. HEAppE provides to the system a better security level by mapping external (LEXIS) user's identity (portal users) to HPC centre identity (HPC users). HEAppE is also used for authentication and authorization to cloud infrastructure like a wrapper, which gives the orchestrator access to the cloud infrastructure. In the architecture, we have an Approval System, which provides the possibility for the HPC centre responsible persons to smartly approve requests for computation resources. The Approval System has other useful features (deployment version, current instance status, numbers of running jobs etc.) to manage HEAppE instance in the HPC centre.

As stated in Section 3 of deliverable D2.1 [1], security has implications almost everywhere in LEXIS architecture. In the diagram, the integration between LEXIS AAI service and other components is represented with dashed lines. The LEXIS architecture analysed from the security point of view is available in Deliverable D4.5 [3].

This section highlights the key components of the LEXIS platform, which are also relevant for the next section on lessons learned. The details of the LEXIS platform deployed by M15 are to be found in Deliverable D3.3 [4].

[1] Note that the LEXIS Front-End as presented here is not the same as the LEXIS Portal Front-end as defined within Deliverable D8.1 [3]. The former term focuses on the broader user experience when interacting with the LEXIS Platform; the latter is a specific software component which runs within a browser as defined in D8.1 Also, the Portal API as defined within D8.1 maps to the LEXIS API Proxy within the LEXIS Services layer in Figure 1.

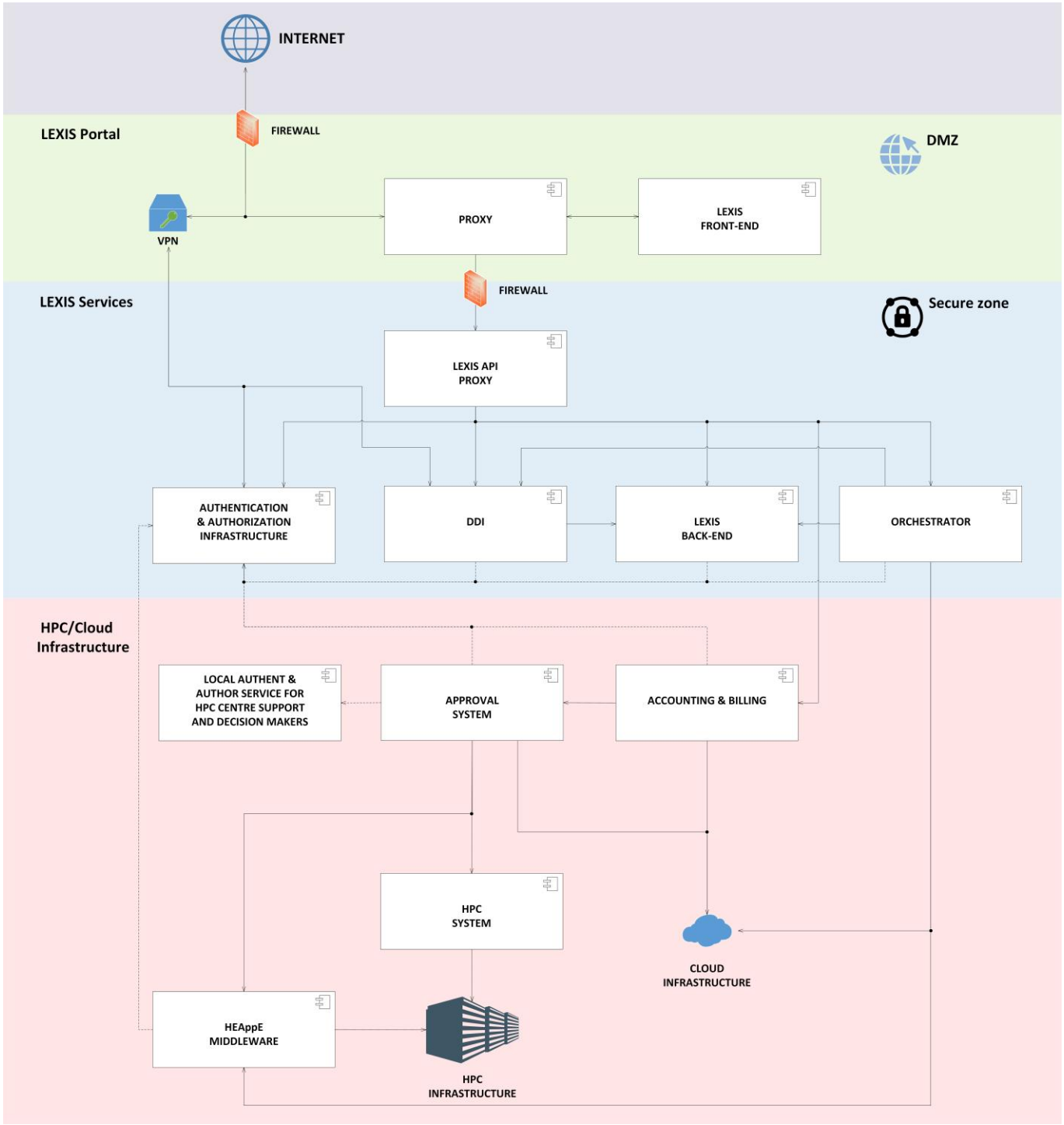

Figure 1: LEXIS Architecture Diagram

2.1 LEXIS PORTAL

The LEXIS Portal is the main entry point to the LEXIS platform: it is designed primarily for the users who do not necessarily have deep experience with working with HPC and/or cloud environments while being provided an easy access to the most advanced capabilities and features[2].

Progress on design and implementation of the LEXIS Portal is ongoing: an initial internal release - R1[3] - was delivered in Q4/19 which provided basic authentication capabilities, had initial user and organization models, and provided some basic integration with the DDI. The primary features required for the R2 release – planned to be delivered M15 (March 2020) - is the ability to deploy basic workflows via integration with the orchestration system (Alien4Cloud) as well as integration with the CYC Accounting and Billing system. Developments relating to the support for an enhanced AAI model are also ongoing, specifically to have greater supports for different authorizations within different services.

The LEXIS Portal comprises of the following components:

- LEXIS Portal Front-End: a rich web client which talks to the LEXIS Portal API,
- LEXIS Portal FE server: a lightweight process which serves the Front-End and supports an OpenIDConnect workflow,
- LEXIS Portal API: an API that interfaces with the Front-End and controls access to the other services within the system,
- LEXIS Dataset Management Interface: a module that communicates with the LEXIS DDI to obtain directory listings, file contents etc,
- LEXIS Workflow Orchestration Interface: a module that interacts with the Alien4Cloud module of the Orchestration Service described below to support workflow deployment and monitoring,
- LEXIS UserOrg Service: a service that stores LEXIS user/organization/project mappings.

All of these components have been developed and are integrated. However, they are not yet functionally complete and will evolve as the project progresses into the R3 and R4 cycles.

More details on the design of the LEXIS Portal components can be found in deliverable D8.1 [5].

2.2 LEXIS SERVICES

LEXIS offers three key services which address the main concerns of working with HPC centres, these are:

- AAI Service, which relies on Keycloak as well as OAuth2, OpenID Connect and SAML standard frameworks to guarantee the entire LEXIS platform's security,
- DDI Service, which is based on iRODS for collecting, managing, storing, retrieving, and providing data,
- Orchestration Service, which relies on Ystia Orchestrator (Yorc) to schedule the execution of tasks on the compute nodes throughout the application life cycle.

The sections below introduce the current development status of these 3 services.

[2] The LEXIS Portal is of course not the only way to interact with the LEXIS platform - it is possible to use standard HPC tools etc. in conjunction with the LEXIS Portal.

[3] WP8 decided to release the LEXIS Portal in four releases - R1, R2, R3, R4 - with increasing levels of functionality.

2.2.1 AAI Service

Technology

LEXIS AAI is based on Keycloak[4] that has been selected earlier during the project execution and described in Deliverable D4.1 [6] on "Analysis of Mechanisms for Securing Federated Infrastructure". This specific core component of LEXIS architecture is in charge of handling the LEXIS Identity (LEXIS users or LEXIS process) and the access (such as LEXIS Portal management access or DDI/WCDA and Orchestrator "list", "read" and "write" access) through authentication and authorization.

RBAC matrix

A specific Role-based Access Control Matrix (RBAC) defining all access permissions needed in LEXIS platform in terms of Role has been designed, refined several times, and finalized. The implementation of this RBAC Matrix in Keycloak is currently ongoing. This implementation is using only one Realm[5] for the whole LEXIS platform and several groups (with attributes) defined at realm level that can be reused in Keycloak "clients" (see Deliverable D4.5 [3] for details).

In LEXIS project, a Keycloak client[6] is a LEXIS service that uses Keycloak to delegate all its resource or data access requests (DDI/WCDA, Orchestrator, Portal etc.). Because Keycloak also allows to define relationships between roles and fine-grained Groups, User Attribute or even specific Security Policy if needed, it enables a very flexible management of LEXIS AAI system. The details regarding how Keycloak was selected as LEXIS AAI solution can be found in Deliverable D4.1 [6].

All different configurations that were previously used in each WP (WP8, WP3 and WP4 for LEXIS services) have been centralized in one Development Keycloak instance that is currently hosted at LRZ on the OpenStack platform. A Proof of Concept has been made to integrate all LEXIS services and the Keycloak-based AAI.

Current state

At the time of writing, the registration of clients has been done which means LEXIS AAI system is able to authenticate user identity, and we are currently finalizing the authorization part which is configured using the RBAC Matrix for setting up user role mappings permissions to specific users.

The deployment method that has been envisioned during the co-design phase (see details in Deliverable D4.1 [6]) will be implemented once the authentication and authorization parts are finalized.

Also, LEXIS to HPC Identity mapping has been described in D4.1 [6]. The only change from the very first stage in the co-design phase is that LEXIS AAI does not handle any IT4I or LRZ user accounts, but only LEXIS user accounts which allows us to:

- Handle federation at (global) LEXIS platform level instead of Authentication and Authorization at local levels,
- Improve security by adding better isolation and improving separation of duties between LEXIS cloud infrastructure and local HPC service provider levels.

[4] Keycloak: https://www.keycloak.org/

[5] A realm manages a set of users, credentials, roles, and groups. A user belongs to and logs into a realm. Realms are isolated from one another and can only manage and authenticate the users that they control.

[6] Clients are entities that can request Keycloak to authenticate a user. Most often, clients are applications and services that want to use Keycloak to secure themselves and provide a single sign-on solution, see https://www.keycloak.org/docs/6.0/server_admin/

2.2.2 DDI Service

The "Distributed Data Infrastructure" (DDI) of LEXIS is a central storage library, providing a unified view of LEXIS data (at least on its central parts) from all participating institutes. The DDI and other storage systems used in LEXIS (see D3.3 [4]) storage libraries are equipped with interfaces (primarily mostly web APIs) to interact with the LEXIS Portal (WP8) and the orchestration solution (WP4). For Authentication and Authorisation, WP3 leverages the solutions developed in WP4, and integrates with the Technology/co-design plans and roll-out procedures defined in WP2.

iRODS[7] has been proposed in LEXIS as a basis for building a DDI compatible with the European Research Data Infrastructure project EUDAT[8]. In M1-M6 of LEXIS project, various iRODS test configurations were deployed and speed tests were performed. First, EUDAT services (B2HANDLE, B2SAFE, B2STAGE) with which the DDI might integrate, were tested and problems identified and resolved. Collaboration with EUDAT – which seems – has turned out fruitful for WP3 and LEXIS as a whole: B2SAFE, B2STAGE and B2HANDLE were successfully deployed as test instances and a production B2HANDLE instance is being set up.

The progress since M6 has focused on definitive solutions for configuring and interacting with iRODS within the LEXIS context, on advances and consolidation in the systems installed, and on surveying of software potentially useful for monitoring the system. After successfully reaching project milestone MS2 (with the main requirement that the DDI be partially installed, in M9), we concentrated in particular on setting up a redundant iRODS/EUDAT DDI core, and integrating it with Keycloak, one of the main building blocks of the LEXIS AAI solution. Problems with this – which could be traced back to the iRODS-OpenID plugin not accepting large tokens as issued by Keycloak (see Section 3.2.2 as well as D3.3 [4] and references therein) – required a significant effort until a workaround was found with moderate modifications in the systems. By M12, the installation of the DDI (iRODS-EUDAT system) was concluded in both centres (IT4I/LRZ), with a few remaining issues being resolved in M13.

The current status of the DDI with references to technical details is summarized in D3.3 [4], Section 3. Hardware systems utilised are described in D3.3 [4], Section 2. To summarise, despite some difficulties, the deployment follows our initial design plans.

2.2.3 Orchestration Service

Orchestration service represents one of the key technological pillars of the LEXIS platform. It provides the features that enable LEXIS users to run their workflows using federated resources available in one or more HPC service providers (starting with IT4I and LRZ).

API

The LEXIS orchestration service is based on the integration of several modules into a functional (see also Deliverable D4.2 [7] — Section 1). Specifically, the service architecture uses Alien4Cloud (A4C) as front-end of Yorc orchestration service. The architecture also contains a monitoring module and a business logic module providing dynamic placement capabilities for the workflows' tasks. Architecturally speaking, orchestration service API is exposed through an API module. The details can be found in deliverables D4.2 [7], D4.3 [8], D4.4 [9] and D4.5 [3] with the respective focus on

- The software implementing the intermediate version of the orchestration system and its installation on the LEXIS infrastructure,
- The data management aspects related to the orchestration,

[7] iRODS: https://irods.org
[8] EUDAT: https://eudat.eu

- The important requirements and constraints needed to design and implement the orchestration system,
- The measures being taken to guarantee different systems (including the orchestration system) within the LEXIS platform security.

As such, the orchestration service architecture has an API sub-module which acts as an end-point for other services and modules foreseen in the LEXIS architecture; indeed, the LEXIS Portal will rely on this API to interact with orchestration service.

Functional modules

Orchestration capabilities are provided by two sub-modules, namely:

- The Yorc server,
- The Alien4Cloud front-end for the Yorc server.

The Yorc server is in charge of executing workflows and to actually select, for each LEXIS task to execute, the set of resources to use (i.e. IT4I infrastructure vs LRZ infrastructure etc.). Since the underlying resources can be those exposed by the HPC clusters and Cloud, the access to the former is mediated by a middleware (HEAppE). While controlling Cloud resources (e.g. OpenStack) can be directly performed by Yorc server, to enable the Yorc server to seamlessly access the HPC clusters, a Yorc plug-in for HEAppE has been developed and integrated. Yorc server is able to directly control Cloud resources (OpenStack) for what concerns the deployment of virtual computing instances. However, in the case of LRZ OpenStack configuration, there is no way to use authentication tokens provided by Keycloak. To overcome such limitation, an OpenStack wrapper integrated in Yorc has been added to delegate authentication to the HEAppE middleware (i.e. the OpenID token received by Keycloak is translated into the one accepted by OpenStack Keystone). Beside computing resources, storage ones are exposed through the DDI. As such, Yorc server also uses the HEAppE middleware to get access to DDI resources.

The Alien4Cloud (A4C) front-end provides the interface for managing workflows (i.e. to define workflows, run workflows etc.). A Go client library has been developed and made available to give access to this interface. A4C is connected to the Yorc server through a plug-in. A4C also implements a catalogue of already available TOSCA[9] workflows and TOSCA components (i.e. TOSCA entities for which orchestration actions are defined — e.g. deploy, run, start, stop etc., and that are used to compose a runnable workflow), while it is possible to add custom components. Typically, WP6 and WP7 LEXIS pilot workflows have specific requirements that required the implementation of additional components to manage their whole execution. LEXIS will also provide generic workflows which are not specific to the LEXIS pilots; users will be able, for example, to provide docker containers as compute entities which can be inserted into predefined generic, sequential workflows.

Connectors to other LEXIS services

Beside these foundational sub-modules, orchestration service also implements the AAI connector to enable interaction with the LEXIS AAI service: a monitoring sub-module to collect metrics used to drive orchestration policies and a business logic sub-module. This latter is designed as a block providing placement services for dynamically allocating LEXIS workflows' tasks (i.e. LEXIS tasks).

At the moment of writing this document, several modules belonging to the orchestration service are in place (i.e. A4C, Yorc, Yorc-HEAppE plugin, A4C go-client library). During the next months of the project, also business logic functionalities will be designed and integrated (this module will also use collected monitoring information), as well as full integration with the LEXIS Portal.

[9] Topology and Orchestration Specification for Cloud, which allows user to specify a cloud application's topology by defining a set of nodes that are connected to other using relationships.

2.3 INFRASTRUCTURE LAYER

In this section, we begin with a brief introduction of the HPC systems and the available cloud resources in the two HPC centres, and then we discuss the installation status of storage system and our research on FPGA usage.

2.3.1 HPC

LRZ and IT4I HPC systems used in the LEXIS project are production-quality systems available independently of LEXIS, and used in shared usage mode together with other users. Details on these systems have been presented in Deliverable D2.2 [2] and updates are provided in Deliverable D3.3 [4].

The systems relevant for LEXIS are the LRZ Linux Cluster and SuperMUC-NG, as well as a DGX-1 machine[10]. The operational infrastructure at IT4I consists of the Barbora, Salomon, and Anselm clusters and a NVIDIA DGX-2[11] machine.

2.3.2 Cloud

One aspect of the LEXIS project is to provide support for heterogeneous HPC/Cloud resources - where cloud resources can flexibly be used for smaller or parallel, but less communication-heavy jobs, e.g. pre- and post-processing and visualization of results (please refer to Section 2 in D3.1 [10] for details). The following cloud resources are available within the HPC centres:

- IT4I deployed a 3-node VMware cluster and another 3-node OpenStack deployment is in progress. This deployment is operated by the LEXIS team and it is not part of regular production at IT4I Therefore it can be reconfigured to fit any need arising from the project.
- LRZ allows shared usage of the LRZ Compute Cloud, a production system based on OpenStack as well, by LEXIS.

Further details on these systems can be found in D3.3 [4] and D2.2 [2] (and references therein).

2.3.3 Data Storage

Both IT4I and LRZ have installed storage system for LEXIS, and the storage provided in IT4I will serve as the main storage. Bull/Atos is developing newer experimental technologies which the project is experimenting with, focused on the usage of Burst Buffers, two of which have been deployed at IT4I and LRZ each. Details on the current status of storage systems used by LEXIS can be found in D3.3 [4], Section 3, where also links to the earlier descriptions in D3.1 [10] and D2.2 [2] can be found.

At IT4I, Ceph storage cluster has been installed as a part of the experimental infrastructure. It provides 120 TB of raw storage through 4 OSD servers and redundant 25 Gbps connection to each. This storage will serve as the main storage for the DDI, as staging storage where needed, and as the main storage for the VMWare and OpenStack deployments which will support the LEXIS platform. The Burst-Buffer servers at IT4I (see below) are connected to the Ceph infrastructure as well.

LRZ has added an experimental storage system for LEXIS, to be used as a back-end for testing purposes (e.g. for the Burst Buffer or the WCDA) and as a scratch space. In addition, LEXIS will rely on shared usage of the several production-quality storage services of LRZ. In particular, 50 TB on LRZ's Data Science Storage (DSS) have been acquired/reserved for LEXIS as a staging area and storage back-end for the DDI (it is also divided into staging area and storage back-end in IT4I). The Burst Buffer systems deployed at LRZ have access to the DSS. While the general

[10] LRZ documentation: https://doku.lrz.de
[11] IT4I documentation: https://docs.it4i.cz

design of storage solutions follows our initial thoughts, the LRZ experimental storage system was added as a component, based on the conclusions from Task 3.1 (survey of available storage systems, see D3.1 [10]).

Burst Buffers are built from arrays of high-end storage devices (NVRAM, SSD) and positioned between the front-end computing processes and the back-end storage systems to speed up the data access in terms of bandwidth and latency. The Bull/Atos SBB (Smart Burst Buffer) solution is based on an innovative and flexible usage of intercepting IO related calls to the glibc[12]. In this way, the end user can define, in the runtime environment of its job, a set of targets (i.e. wildcard file path, directory etc.) to be managed with the Burst Buffer. Then, all IO related to these targets are transferred to the Burst Buffer which stores them in its low latency and high throughput storage (RAM, NVMe etc.). In parallel, data stored in the Burst Buffer are asynchronously destaged to the end parallel file system initially targeted by the intercepted IOs. More details on the SBB architecture and usage could be found in Section 3.3.3 of the deliverable D3.3 [4].

Bull/Atos Smart Burst Buffer was initially designed for HPC (High Performance Computing) clusters using POSIX compliant parallel file system (e.g. LustreFS) and RDMA (Remote Direct Memory Access) protocol over infiniband interconnect. However, to support LEXIS main use cases (please check deliverable D2.2 [2] in Section 3.2.3 for details) the following new features are under the development:

- The support for NFS (Network File System). The Bull/Atos Smart Burst Buffer solution uses the filehandle of the file to identify it between the client and the Burst Buffer server. However, NFS does not expose the file's filehandle, and therefore the filehandle approach had to be expanded on information available on NFS, such as inode. This new feature is currently under validation.
- The support of the TCP/IP protocol, instead of RDMA, for transferring data between the client and the Burst Buffer server. The support of this first protocol is currently under the development.

Thanks to NVMe-oF (NVME over Fabric), the Bull/Atos SBF (Smart Bunch of Flash) solution allows one node of a job to have access to an efficient exported data storage device. As for the Smart Burst Buffer, the Bull/Atos SBF solution is based on the RDMA protocol over infiniband interconnect. To support the LEXIS infrastructure based on ethernet network and virtual machines, a Proof of Concept using the RoCE (RDMA over Converged Ethernet) fabric and its software implementation is currently the under development at LRZ.

2.3.4 FPGA Accelerator

During the co-design phase, the use of FPGAs in the LEXIS project has been investigated from several angles, and not only from a high performance computing point of view. Data processing acceleration actually demonstrates real opportunities and could complement the use of Burst Buffers (please check previous paragraph as well as Deliverable D2.2 [2] in Section 3.3 for details). To be more specific, data management tasks such as data encryption, compression, rasterization or format conversion, could more easily take advantage of FPGAs than the computing software (for several reasons explained in D2.2 [2]).

We investigated this technology with a special focus on the usage of Burst Buffers beyond this niche of data access speed-up, i.e. how they could be used for accelerating performance in other areas where the ability to handle huge volume of data with high speed and low latency is a key.

Therefore, our working efforts have focused on the following themes:

[12] The GNU C Library, commonly known as glibc, which provides the core libraries for the GNU system and GNU/Linux systems, as well as many other systems that use Linux as the kernel, see https://www.gnu.org/software/libc/

- Analysis of the DDN/IME (Infinite Memory Engine) architecture together with its integration within an HPC system,
- FPGA acceleration of the following potential uses: in-situ analysis, streaming applications (e.g. compression), application pre-processing and/or post-processing etc.

Our work focused mainly on the hardware integration of FPGA including the analysis of power supply, interconnection protocol, device set up (clocking scheme, programming etc.) which lead to the selection of a range of FPGA candidate (e.g. Altera/Stratix10) and their associate boards (e.g. Bittware, ex-Nallatech, 520N card like the one installed at IT4I).

Next steps will be more oriented towards implementation.

2.4 DEPLOYMENT STATUS SUMMARY

For getting an overall understanding, the deployment status of the components introduced above is summarised in Table 1.

COMPONENT	DEPLOYMENT STATUS	SEE SECTION ABOVE
LEXIS Portal	R2 is to be released in M16	2.1
AAI Service	Standalone mode done; Aiming at Cross Datacenter Replication Mode	2.2.1
DDI Service	iRODS-DDI installation and federation has taken place; first version of DDI APIs usable; EUDAT integration in progress	2.2.2
Orchestration Service	Several modules belonging to the orchestration service are in place; Business logic functionalities will be designed and integrated	2.2.3
HPC	Basic integration of existing LRZ and IT4I petascale systems has been completed	2.3.1
Data Storage	Storage systems installed in both LRZ and IT4I, and Bull/Atos is developing newer experimental technologies	2.3.2
FPGA Accelerator	Investigation on use cases done and one card installed; Implementation to be done	2.3.3
Cloud	3 node VMware cluster deployed; 3 node OpenStack deployment is in progress	2.3.4

Table 1 Deployment Status of LEXIS Key Components

3 LESSONS LEARNED ON TECHNICAL DEVELOPMENT AND INTEGRATION ACTIVITIES

The co-design phase is progressively being replaced with activities aiming at identifying and documenting the most significant lessons learned (LL) resulting from the co-design activities until the official end of the project.

As introduced in the previous section, during the functional and technical co-design phase [M1-M15], a set of key components have gone through the process from investigation to implementation and integration. Some of them presented good practices, differences between foreseen objectives and achieved ones, and challenges during the process.

Besides on pure technology aspects, we also identified potential ones in the way organizations will have to interact, cooperate, and design consistent processes to bring the platform to production and accept external users in compliance with local and European regulations.

In this section, we present our LL starting with the adopted LL methodology.

3.1 LL METHODOLOGY

The section introduces LL methodology by coming up with the 3 questions:

- Why do we need LL?
- What information was recorded?
- How LL were organized?

And then the exact collection approach is illustrated.

3.1.1 Why do we need LL

Capturing lessons learned is an essential activity of any project throughout the project life cycle. We learn from project failures as well as successes. If we do not learn from failures, we may reproduce similar situations in the future. If we do not learn from success nor maximize it, we may miss opportunities to implement good practices to complete existing and future work [11]. In the context of LEXIS, which aims at generating valuable outcomes and improving the efficiency and quality of services, the specific objectives of LL are:

- Improving implementation of newly proposed LEXIS approaches, e.g. LEXIS Authentication and Authorization Infrastructure (AAI), LEXIS Distributed Data Infrastructure (DDI),
- Preventing or minimizing risk of failure,
- Better planning later project phases.

3.1.2 What information was recorded

In general, LL have the form of documented information presenting feedbacks during technology deployment throughout the project life cycle. To be more detailed, the information contains:

- Best practices,
- Differences between foreseen objectives and achieved ones,
- Challenges/Issues during the project.

To meet LEXIS objectives, significant LL were identified and collected. The significance was estimated from the following aspects:

- **Effort:** Was high workload taken on?
- **Novelty:** Did we make innovations or any breakthroughs?

- **Impact:** Would the LL impact other or the following activities of the project, or provide a guidance on solving other problems?

3.1.3 How LL were organized

After knowing how significant LL were identified, next question comes out: How LL should be organized?

From the analysis of outcomes of Deliverable D2.1 [1] and D2.2 [2], LL were divided into the following categories:

Technology related LL

These include:

- Pilot application related LL, including:
  - Aeronautics,
  - Tsunami and Earthquake,
  - Weather and Climate.
- Technology related LL, including:
  - AAI – harmonization of identity management in 2 federated HPC providers,
  - DDI – delegation of DDI security operations to AAI,
  - Orchestrator – workflow and resource management,
  - FPGAs (AND GPUs) – usage for LEXIS: compression, encryption, conversion, SMP jobs, etc.
- General – harmonisation of technical vocabulary**.**

Organization related LL

These include:

- General – define LEXIS cloud delivery model,
- General – federation challenges**.**

Other related LL

- General – Release management challenges.

LL content characterisation

To guarantee the consistency and relevancy, the necessary components of LL were defined as follows:

- **LL Description:** Brief summary of the findings and recommendations for correcting the findings,
- **Relevant activities:** What activities led to the LL?
- **Challenges or issues:** What are the challenges or issues that were or will be faced during the implementation activities?
- **Proposed approaches:** What are the envisioned solutions?

3.1.4 Approach to insert or update LL

Every time a new insertion was done by creating a new table as follows and filling the fields in the corresponding section.

Every two weeks, a reminder email was circulated, and the document was checked and analysed by LL task leader weekly. The clarification of certain LL was communicated through emails or telco meetings. The status and issues of LL were discussed in the telco meeting biweekly.

3.2 TECHNOLOGY RELATED LESSONS LEARNED

In this section, we present LL on technology aspects.

3.2.1 AAI – Harmonization of identity management in 2 federated HPC providers

To ensure secure management of data within a federation of service providers, we need to federate identity and access management using up-to-date standards. As stated in the LL in Table 2, we found different security policies in the 2 HPC centres. The best way to connect the two parties which do not trust user accounts the same way, nor secure the use of schedulers the same way, is through the HEAppE middleware (and abstraction layer), developed in IT4I and adapted to the production environment in LRZ, too.

Author	Date	LL title	LL Brief Description
Marc Levrier	12/12/2019	HEAppE common security middleware	Solution of the different security policies in the LEXIS federation of several HPC providers.
Questions			**Answers**
What activities led to this LL?			Co-design activities aiming at finding a common model and implementation to handle identity management and authentication in a coherent way between: IT4I and LRZ HPC providers and ECMWF data provider, Production HPC parts in both LRZ and IT4I centres, Production Cloud part in LRZ and LEXIS project Cloud part in IT4I.
What are the challenges or issues?			It turned out IT4I and LRZ have incompatible security policies regarding on both topics. The consequence is that standard components in place in both organizations (LDAP, OpenStack and their relationship with HPC environments) can neither communicate with nor trust each other. LEXIS federation cannot rely on these components directly.
What are the envisioned solutions?			We plan to have both HPC providers trusting HEAppE software (developed in IT4I and to be deployed in LRZ). In this solution, the HEAppE middleware abstracts the distinct policies in place at both sites and is in charge of mapping LEXIS user accounts to hidden accounts on each site.

Table 2 LL on HEAppE common security middleware

3.2.2 DDI – Delegation of DDI security operations to AAI

The LEXIS Distributed Data Infrastructure has the objective of providing a unified view on LEXIS data for all users and from all systems in the participating computing / data centres. As a framework to build the infrastructure with, we chose the “integrated Rule-Oriented Data System” (iRODS), a solution for distributed data management. This system integrates well with the European Collaborative Data Infrastructure (CDI) EUDAT (as it is the basis of EUDAT-B2SAFE), and has also been used in earlier projects by the LEXIS participants.

As a federated, synchronised AAI is used in LEXIS. It had to be made sure that the DDI delegates authentication and partially authorisation/access-granting to this AAI. This however led to a problem, as the iRODS-OpenID auth plugin proved incapable of dealing with the long OpenID tokens from Keycloak servers, which are used as AAI solution in LEXIS. In order to resolve this, we had either to i) avoid passing the long tokens inside the iRODS system, or to ii) change the AAI or DDI solution. It turned out that, after relatively significant programming work, passing only a token hash inside iRODS was possible, allowing us to implement the system as envisaged. This result allows for the collaboration with EUDAT as initially aimed for.

The challenge here demonstrated that innovative software is not always perfect, and unforeseeable problems, especially in integration, can result in. This may in particular be true for software like iRODS, which comes from an academic research context. Systems with larger industrial consortia behind may be an alternative with smaller probability of such problems, as broad commercial application is likely an implicit test for practically every integration scenario. However, also systems with strong industrial support do often need customisation for a smooth usage experience (e.g. OpenStack in the LRZ production Compute Cloud). To summarise, we have found a good solution for the DDI with iRODS/B2SAFE, and the lesson learned is described with some technical details in Table 3.

Author	**Date**	**LL title**	**LL Brief Description**
Stephan Hachinger, Mohamad Hayek	**11/04/2019**	**Development effort needed to delegate DDI security operations**	As LEXIS is based on a federation of computing and data centres, it has been decided to design a common Authorization and Authentication Infrastructure (AAI) and to use this AAI when granting access to LEXIS services. The LEXIS Distributed Data Infrastructure (DDI) thus has to delegate authentication and parts of authorisation to the central AAI. This required some contribution to the iRODS open source community.
Questions			**Answers**
What activities led to this LL?			Connecting LEXIS DDI (based on iRODS), and our AAI (based on the state-of-the-art OpenID Connect (OIDC) and Keycloak standards), we realized iRODS support of OIDC has some maturity problems (unstable/unfinished plugin). Specifically, the plugin has e.g. an internal memory limitation to the token size, which Keycloak (the LEXIS AAI solution) happens to exceed with its long tokens.
What are the challenges or issues?			iRODS was chosen for making the LEXIS DDI compatible with EUDAT (B2SAFE) and after significant testing adopted as definitive LEXIS DDI technology in the 2nd half of 2019. Because of considerable investment in iRODS as DDI system, and progress, as well as LEXIS time passed, changing to another solution is not possible. The timeline for having the iRODS OIDC plugin stable from upstream is unknown.

	Therefore, we are left with iRODS and the current plugin problems.
What are the envisioned solutions?	The current iRODS-OpenID plugin and auth broker server will be adapted for LEXIS such that: Between different iRODS components, token hashes are transferred instead of tokens, making it possible to keep the original buffer variables. The OpenID authentication broker stores both token hash and token, such that the full token can be accessed when needed, requesting it from the broker. The auth workflow of the iRODS-OpenID system will be made to work again, considering the decisions above. Note: Another way to handle the problem of the size of the token is to increase memory size for transferring the token by "chaining memory blocks" (the first memory block contains pointers to other memory blocks). This technical implementation avoids storing both token and its hash, but it is much more complex (memory allocation/deallocation and chaining mechanisms) to implement and make it production ready.

Table 3 LL on development effort needed to delegate DDI security operations

3.2.3 Extending orchestrator to cover pilot workflows specific requirements and constraints

A set of user-level criteria and metrics are used to drive the allocation of resources in the federated HPC/Cloud environment. During the investigation, a set of LL appeared on numerous aspects, e.g. supporting urgent computing, heterogeneous infrastructure calls for an abstraction level, the needs to minimize the supported standard to serve complex workflows and to enable dynamic orchestration etc. The details are presented in Table 3 - Table 9.

Author	Date	LL title	LL Brief Description
Emanuele Danovaro	12/16/2019	Realtime constraints & orchestration	Weather forecasts are highly time sensitive, so most operational services rely on dedicated computing resources and limit the computational needs to fit in the available resources. Some of the LEXIS WP7 workflows are targeting analysis and forecast of extreme weather events, so emerged the need for low-latency on-demand numerical weather prediction (NWP) regional downscaling and related domain and namelists definition.

Questions	Answers
What activities led to this LL?	During the analysis of the WP7 workflows and related requirement on the Workflow orchestrator, emerged a requirement of on-demand, low-latency regional downscaling of NWP to better assess the impact of extreme events.
What are the challenges or issues?	Challenge #1: No planned system for fully automated domain definition, name list creation and experiment execution. Challenge #2: No support for urgent supercomputing (so far).
What are the envisioned solutions?	For Challenge #1: Development of a REST API for NWP regional downscaling. Input parameters are domain extents (coordinated of the bounding box) and required resolution. The system will create the name list with appropriate domain nesting and buffer zones, will extract initial & boundary conditions from WCDA and trigger the execution. For Challenge #2: The availability of a distributed computing infrastructure, with a rich portfolio of computing resources, enables a new approach in which the computing needs are tailored to the user request and the resource allocation is fully managed by the orchestrator that may submit the time critical jobs on multiple computing resources and, as soon as the computation starts on one of them, releases all the other allocated resources.

Table 4 LL on realtime constraints & orchestration

Author	Date	LL title	LL Brief Description
Emanuele Danovaro	**10/31/2019**	**API for user access to meteorological data**	Meteorological archives available to LEXIS users are able to support two really different data types: observations, usually encoded as time series, and model outputs that are significantly larger (ECMWF is currently producing 200 TB/day). We need a system to semantically query such large archives and support user needs.
Questions			**Answers**

What activities led to this LL?	In the development of complex workflows exploiting meteorological forecasts, emerged the need for efficient management of large meteorological archives, complementing DDI services.
What are the challenges or issues?	We are dealing with petabyte-scale archives, guaranteeing efficient data access is a challenge, and we have to deal with different meteorological data models.
What are the envisioned solutions?	We decided to build a domain specific index, associating meteorological metadata to data frames, to guarantee efficient data access on such large archives.

Table 5 LL on API for user access to meteorological data

Author	**Date**	**LL title**	**LL Brief Description**
Emanuele Danovaro	**10/31/2019**	**WCDA portability on LEXIS data stores**	LEXIS has a distributed heterogeneous infrastructure, with different data store architectures in LRZ and IT4I. The data management layer in the WCDA has to adopt different backend to efficiently exploit available resources
Questions			**Answers**
What activities led to this LL?			Development of WCDA and analysis of the LEXIS computing resources
What are the challenges or issues?			LRZ has deployed an experimental storage system, offering full POSIX data system, while IT4I is going to deploy a ceph/rados data system.
What are the envisioned solutions?			We introduced an abstraction level (catalog/store) to manage the heterogeneity and guarantee performance portability across sites. The Catalog is going to handle the metadata index, and will be shared among sites, while the data management back-end will be optimized for the available computing facilities.

Table 6 LL on WCDA portability on LEXIS data stores

Author	**Date**	**LL title**	**LL Brief Description**
Emanuele Danovaro	**10/31/2019**	**Support NetCDF in the WCDA.**	Serving complex workflows, we had to extend the supported standard (adding NetCDF-CF), to minimize the impact on the workflow applications.

Questions	Answers
What activities led to this LL?	From the analysis of meteorological models exploited by LEXIS WP7 workflows, we identified the need to handle two different data formats (GRIB and NetCDF) commonly adopted as meteorological model output. GRIB is widely adopted by global NWP models, while NetCDF is adopted by some of the most common mesoscale NWP models.
What are the challenges or issues?	While a high-level analysis of GRIB and NetCDF data structure is largely similar, metadata management is completely different and requires careful management, in order to guarantee efficient and seamless data interoperability.
What are the envisioned solutions?	NetCDF is a very general data container, so we restricted to a well-defined subset, based on Climate and Forecast (CF) Metadata Conventions, and decided to embed in the WCDA a conversion layer, able to decompose a NetCDF-CF file in the relevant data fields and store them as GRIB data messages.

Table 7 LL on support NetCDF in the WCDA

Author	Date	LL title	LL Brief Description
Alberto Scionti	**01/13/2020**	**Management of WP6 and WP7 workflows**	After a detailed analysis of the WP6 and WP7, additional requirements for the Ystia orchestration layer emerged. Concerning the WP6, the workflow is oriented to urgent computing: • It is composed of multiple sub-workflows. • Management of deadlines for triggering job cancellation (i.e. multiple jobs are launched and when the deadline is reached only the one that completed is kept, the others are killed). Concerning WP7: • The workflow is composed by multiple sub-workflows. • Triggering the launch of tasks when a given deadline is reached.
Questions			**Answers**
What activities led to this LL?			Advancing on WP6 and WP7 pilot workflows modelling on the Ystia orchestrator (TOSCA + HEAppE Command

	Templates); a dedicated F2F meeting has been organized in November 2019 to define how to finalize their porting to Ystia orchestrator.
What are the challenges or issues?	WP6/WP7 workflows are complex (multiple sub-workflows with specific requirements; e.g. running multiple tasks in parallel on different infrastructure at the same time and, once the first tasks complete then kill the others; deadlines etc.) and they need extension of the Ystia orchestrator to manage their specific requirements.
What are the envisioned solutions?	The Ystia orchestrator (Yorc and A4C) is extended to include components (TOSCA) for correctly expressing such requirements (e.g. managing deadlines to trigger new tasks or killing existing ones etc.) in the workflows definition.

Table 8 LL on management of WP6 and WP7 workflows

Author	**Date**	**LL title**	**LL Brief Description**
Alberto Scionti	**01/13/2020**	**Orchestration interaction with other components**	Orchestration Service interaction with DDI layer, dynamic task placement and workflow checkpointing
Questions			**Answers**
What activities led to this LL?			Developing an API to interact with the DDI. Dynamic tasks placement is also foreseen as a feature to be integrated in the orchestration layer; as such a mechanism to extend Ystia has been defined. Also, LEXIS component must be configured to ensure high-availability of the service, by leveraging specific set up and configuration of the architectural components.
What are the challenges or issues?			Extending Ystia to manage task placement in a dynamic manner (dynamic placement function) and interaction with DDI (using DDI API). Workflows also require a mechanism for supporting checkpointing.

What are the envisioned solutions?	From WP3, an initial proposal of how to implement the DDI API for interacting with the orchestrator has been proposed. The Ystia orchestrator will be extended by delegator component, which delegates to an external service (dynamically) to select the location for running tasks. Finally, Ystia components should also be deployed in a HA-configuration in order to ensure availability of the Orchestration service.

Table 9 LL on orchestration interaction w/ other components

3.2.4 Portal design and implementation

The first LL below shows the problem with traditional software development approach. While the linear and sequential development is well structured with clear deadline and documentation, it often lacks flexibility. The increasing endorsements with Agile Methodology may account for mitigating the issues. Many unforeseen problems come during software development, especially when dealing with immature plugins. So, to allow more flexibility for changes, working iteratively by evaluating and adding changes incrementally is a good choice in future case (Table 10).

The second LL provides the evidence to the issues caused by the selection of technology and the strategy of software development which hinders the smooth development as planned (Table 11).

As introduced in LL titled "Development effort needed to delegate DDI security operations" (Table 3), the selection of software is a trade-off among different factors. In addition to the factors mentioned in that LL, while the entire solution is composed by different modus and they all interplay, the familiarity or experience from all team members should also be accounted.

Author	**Date**	**LL title**	**LL Brief Description**
Seán Murphy	**01/13/2020**	**Portal Design needs to be iterative**	The design of the WP8 Portal has been evolving with deeper understanding of requirements and capabilities of underlying technologies; iterative approach required
Questions			**Answers**
What activities led to this LL?			Design and implementation of the LEXIS Portal
What are the challenges or issues?			A number of issues arose during the development of the portal to date which were unexpected, many of which related to security and authentication (e.g. iRODs/OpenID support having limitations, A4C using SAML rather than OpenID). Similarly, the external context is changing dynamically (e.g. development of EuroHPC, new service offerings from cloud providers, new use cases identified), meaning that requirements are always evolving.

What are the envisioned solutions?	The solution - as indicated in the title of this lesson - is to maintain a flexible implementation modus which permits changing the design somewhat as the work develops.

Table 10 LL on portal design needs to be iterative

Author	**Date**	**LL title**	**LL Brief Description**
Seán Murphy	01/13/2020	Diverse background and skillsets of team resulted in much knowledge sharing within portal design process and technology choice selection.	The LEXIS WP8 team has diverse skills ranging from different types of development backgrounds and familiarity with different technologies as well as deep knowledge of HPC context. For this reason, the initial design and technology choice decisions required significant exchange of opinion to determine approaches to project implementation which provided the right compromise between the partner skill sets, the need to collaborate/interwork and not supporting too many technologies.
Questions			**Answers**
What activities led to this LL?			Design and implementation of the LEXIS Portal
What are the challenges or issues?			A key challenge has been that the development team has different backgrounds and needs to work with a diverse set of somewhat experimental tools, meaning documentation is sometimes wanting and/or support is not readily available. Further, the technology choices for the implementation were a compromise which not all team members had experience with.
What are the envisioned solutions?			The solution is time and experience to get all team members up to speed with the technologies, facilitating effective knowledge transfer, and to implement basic functionalities.

Table 11 LL on diverse background and skillsets of team

3.2.5 General – Harmonization of technical vocabulary

Author	**Date**	**LL title**	**LL Brief Description**
Marc Levrier	12/16/2019	Harmonize technical vocabulary	Need to solve ambiguities in how we name technical objects depending on the context.
Questions			**Answers**

What activities led to this LL?	Co-design sessions manifested some misunderstanding during technical conversations involving teams operating at different levels and coming with various kinds of expertise. Some topics, like orchestration, manipulate lots of abstract concepts which are not only difficult to understand but sometimes use names that mean something else in other components of the LEXIS architecture or in other domains such as the project organization itself like: • **Application** (means a single scientific software for pilot teams and a bundle of software and system middleware to be deployed and run together for the orchestrator), • **Workflow** (similar to the previous point), • **Project** (may refer to different objects between portal/billing, HPC providers and OpenStack/Cloud layers), • **User** (means a real/human's public account in the LEXIS cloud layer, and a technical/hidden/anonymous user in the HPC layer), • Etc.
What are the challenges or issues?	The issues are that people involved in co-design come with different backgrounds (by definition) and when vocabulary is ambiguous, its dramatically slows down mutual understanding and lead to misconceptions. This can not only impact internal and external documentation but our APIs as well. For instance, the notion of *project* is radically different in the LEXIS / Open Call world from the OpenStack API. Same as for *applications* or *workflows* that may appear as API objects/endpoints in different places (like A4C API and scientific BPMN descriptions) but mean different things. We must prevent ambiguous and unclear endpoint names, especially if we plan to expose them externally.
What are the envisioned solutions?	Write a little lexicon (only a few objects are a problem) after having agreed widely on how to name these objects and what they refer to in each context or layer.

Table 12 LL on harmonize technical vocabulary

3.3 ORGANISATION RELATED LESSONS LEARNED

In the federation process, though the HPC providers have similar business models and usages, there are other factors - security policies, technology and exploitation processes to be considered, which are demonstrated in Table 13 and Table 14.

3.3.1 General – LEXIS cloud delivery technical and methodological constraints

Author	Date	LL title	LL Brief Description
Marc Levrier	28/01/2020	LEXIS cloud delivery technical constraints	What technical and methodological trade-offs must we accept? The LEXIS co-design phase revealed the challenges in providing full automation. Modern and cloud-native technology, provisioning and ingesting templates and specific developments in the LEXIS Portal providing solutions.
Questions			**Answers**
What activities led to this LL?			The co-design activities related to proposing cloud services for complex HPC and data management workflows, showed that even though state-of-the-art technology will greatly help and allow breakthroughs, full automation and self-service will not be possible. Another interesting aspect is that the granting of resources is legally framed and ruled in a way (e.g. open calls) and come with a mandatory process that we need to take into account in LEXIS.
What are the challenges or issues?			Challenge #1: The LEXIS co-design phase revealed that even though both IT4I and LRZ are equipped with cloud-ready IaaS platforms (OpenStack), a distributed data management system (DDI derived from EUDAT), an AI & HPC orchestrator (Yorc) and a dedicated web portal, technology may not be enough to overcome some structural, legal and security constraints that are specific to state-funded HPC centers and quite orthogonal to the cloud delivery model (self-service and assuming simple application environments). Challenge #2: In the LEXIS case, applications are so complex to deploy that standard cloud provisioning automation cannot be entirely handled by the client project member themselves. Consequently, some of the turn-key and automation promises of the cloud cannot be applied in a context such as LEXIS. Challenge #3: The way compute power is to be granted and delivered to a scientific project that follows strict legal rules and processes that include manual / human arbitration, administration and manual tasks. This induces significant

	administration / paperwork which does not match the traditional cloud models, and as such, traditional cloud practices don't include such concepts.
What are the envisioned solutions?	Challenge #1 is being handled adding modern and cloud-native technology and practices that most HPC centers have not deployed yet on top of cloud / IaaS environments such as OpenStack: end-user web portal for application and data management, tracking and billing, single-sign-on features, remote visualization and overall service management flexibility using REST APIs. Challenge #2 is being handled by providing complex application provisioning and execution templates ingested by the orchestrator and that new projects can start from to accelerate their adoption of the LEXIS platform. In addition to this, professional build and run support services will be organized to help the customers take this step. Challenge #3 is being handled as specific developments in the LEXIS Portal to make the open call related tasks easier and traceable directly from the LEXIS Front-End.

Table 13 LL on LEXIS cloud delivery model

3.3.2 General – Federation challenges

Author	**Date**	**LL title**	**LL Brief Description**
Marc Levrier	**28/02/2020**	**HPC provider federation challenges**	Facing the road blocks in seamlessly joining the forces of 2 or more HPC service providers
Questions			**Answers**
What activities led to this LL?			The LEXIS co-design phase revealed that even though IT4I and LRZ had the same HPC service provider business model and similar usages, their security policies, technologies, and exploitation processes were different enough to challenge the ability to federate them.
What are the challenges or issues?			Existing working methods, security rules and technology choices differ from one federated HPC service provider to another. In the LEXIS context, they were nearly impossible to anticipate in details at the time of project submission to the European Commission (for confidentiality and technical reasons). It took several months of information sharing and in-depth study to have a clear inventory of those differences. Their impact on the solution and its flexibility is however significant.

	Some differences that challenged us were: • Incompatible identity management and authentication policies (detailed in Section 3.2.1), • Different types of HPC job schedulers, • Different flexibility level granted for LEXIS cloud activities (in terms of OpenStack system administration privileges), • Different types of backend storage technologies, • Different grant / legal processes.
What are the envisioned solutions?	The first 3 items are to be handled using the HEAppE middleware developed in IT4I and deployed in both centers, to hide this security management complexity and abstract the Slurm, PBS and LSF job schedulers present on each site. The storage backend inconsistency issue was solved at iRODS level (main building block of LEXIS DDI) which accepts many different types of backends. The last point has required a dedicated implementation in the LEXIS Portal to make it transparent from the user's standpoint

Table 14 LL on HPC provider federation challenges

4 SUMMARY

After 15 months of active co-design, system integration involving the selected technologies has been gradually put into place. A set of achievements have been made as shown below:

- R2 release of LEXIS Portal supporting deployment of basic workflows is to be released in M16,
- About LEXIS AAI:
  - A specific RBAC Matrix (defining all access needed in LEXIS platform in terms of Role) has been defined and finalized,
  - All different configurations that were previously used in each WP have been centralized.
- About orchestration: production ready orchestration components have been deployed and connected to the existing system environment such as OpenStack and the HEAppE middleware for which a plugin has been developed,
- By M12 the installation of the DDI (iRODS-EUDAT system) was concluded in both centers (IT4I/LRZ). Specifically, the difficulty mentioned in Section 3.2 (D2.2 [2]) on the integration of IRODS with the Keycloak-based LEXIS AAI has been overcome. And an interesting lesson learned is based on it (see Section 3.2.2),
- The successful installation of various hardware systems including Ceph storage cluster, FPGA card etc.

Though there is always challenge in trying new solutions and merging the diverse technologies mentioned above, it also means LEXIS produces innovations and breakthroughs. Section 3 records a set of LL which provided initial feedback to the technology selection and deployment. Through summarising the LL, we can see:

- During the deeper analysis on pilot workflows, new requirements appear which need the fast adoption of new advanced technology and solutions,
- The HPC providers' incompatible policies challenge the federation process, which takes considerable efforts,
- Besides more technique issues, the legal and security aspects are not negligible.

While the technology deployment is ongoing, the effectiveness of the solutions proposed in LL will be tracked continuously.

REFERENCES

[1] LEXIS Deliverable, *D2.1 Pilots needs / Infrastructure Evaluation Report.*

[2] LEXIS Deliverable, *D2.2 Key parts LEXIS Technology Deployed on Existing Infrastructure and Key Technologies Specification.*

[3] LEXIS Deliverable, *D4.5 Definition of Mechanisms for Securing Federated Infrastructures.*

[4] LEXIS Deliverable, *D3.3 Mid-Term Infrastructure (Deployed System Hard/Software).*

[5] LEXIS Deliverable, *D8.1 First Release of LEXIS Portal (will include report).*

[6] LEXIS Deliverable, *D4.1 Analysis of Mechanism for Securing Federate Infrastructure.*

[7] LEXIS Deliverable, *D4.2 Design and Implementation of the HPC-Federated Orchestration System - Intermediate.*

[8] LEXIS Deliverable, *D4.3 Definition of Data Access Priority, Analytics Policies, and Security Assessment.*

[9] LEXIS Deliverable, *D4.4 Definition of workload management policies in federated cloud/HPC environments.*

[10] LEXIS Deliverable, *D3.1 Local Storage Solutions Report.*

[11] S. F. Rowe and S. Sikes, “Lessons learned: taking it to the next level,” in *PMI® Global Congress*, North America, Seattle, WA, 2016.