The ATLANTIS System Architecture

ATLANTIS Public Report Nr. 5	
Project:	ATLANTIS – AuThoring tooL for indoor Augmented and dimiNished realiTy experIenceS
Website:	http://atlantis-ar.eu
Author(s):	Richard Whitehand, Per Ström (UP), Julia Parinova, Robert Huemer (ROOM), Vladimiros Sterzentsenko, Georgios Albanis, Nikolaos Zioulis, Dimitrios Zarpalas (CERTH), Werner Bailer (JRS)
Publication date:	2021-05-11
Version number:	1.0
Abstract:	This report describes the initial system architecture of ATLANTIS and the underlying design considerations.

Introduction

This report presents an overview of the initial ATLANTIS system architecture, and the underlying design considerations.

The architecture is expected to be refined after the first prototype system has been evaluated. The feedback from these evaluations, as well as any possible new or extended requirements will inform the revised architecture.

The system architecture has been defined based on the analysis of user and technical requirements. The initial version of the architecture reflects the current assumptions about the functionalities of the technical components and the way they are integrated in the user interfaces. The architecture includes the technical components that will be developed in the first iteration of the ATLANTIS system as well as those envisaged to be realised in a later stage.

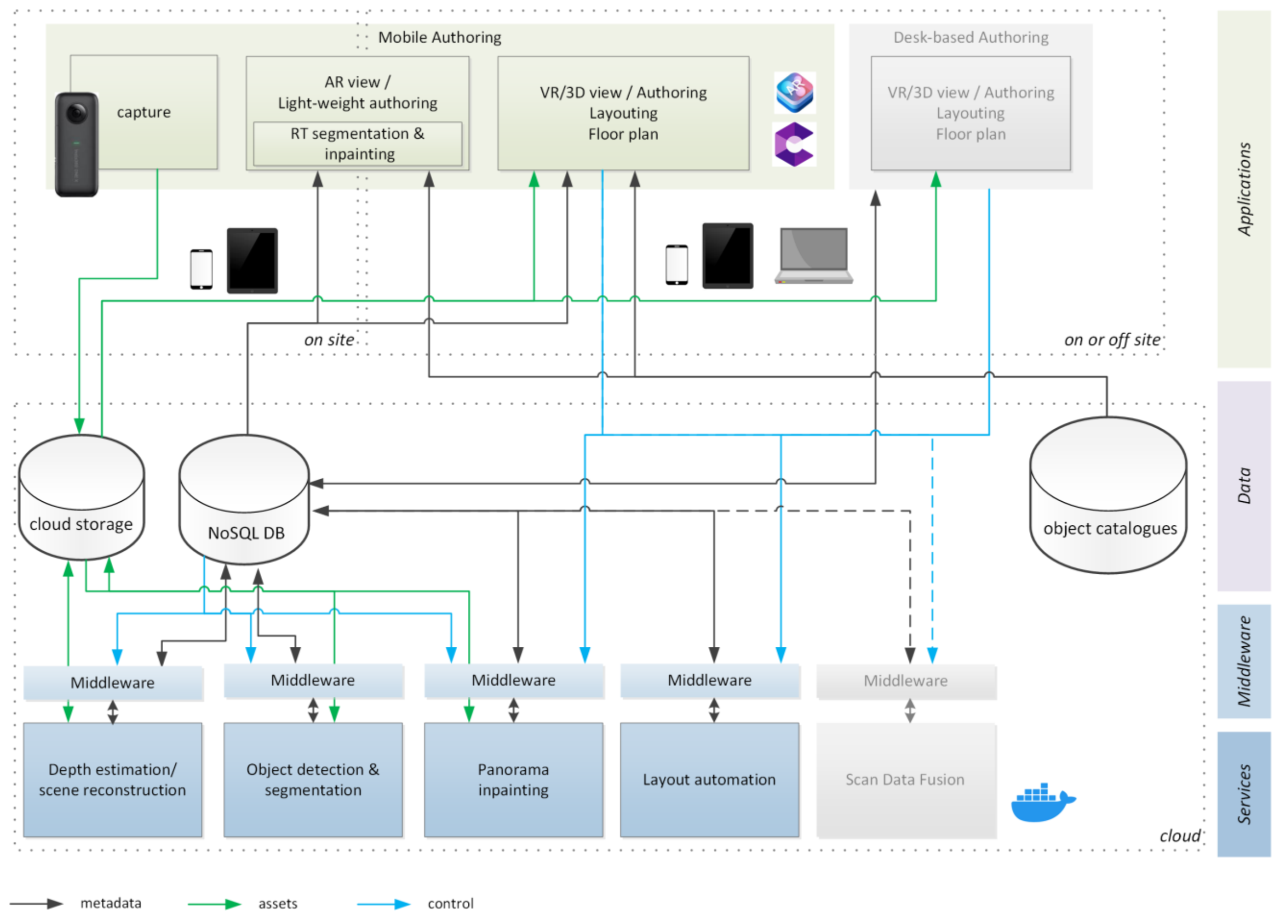

Figure 1. ATLANTIS system architecture.

Figure 1 shows an overview of the initial architecture (the components in gray are placeholders for envisaged components not included in the first iteration, and their specification and development will be defined in the next iteration). It is organised into four layers:

The **Applications** layer represents the user facing applications of the ATLANTIS system. Initially, those will be realised as mobile applications (running on mobile phones, tablets) used on-site (i.e. in the room being planned) or off-site (e.g., in the room next door, in the office, at a sales outlet). In addition, there could be desk-based applications, such as the PC of a designer or sales professional, or a specific device such as a kiosk-based system at a store (to be decided later whether those will still be realised within the project duration). Note that we use the terms “mobile” and “desk-based” not strictly in a literal sense, but rather to denote portable devices running a mobile OS with native apps and touch-based interaction vs. devices that would use web-browser technology and typically have additional I/O capabilities (e.g., keyboard and mouse). The ATLANTIS service-oriented architecture was designed so that it can accommodate all these options for its application layer as the interfacing with the application layer is loosely coupled.

Both mobile and desk-based applications include authoring and viewing functionalities. The capture and AR functionalities are only included in the mobile application, as they need to be used on site. The other functionalities overlap between the two types of applications. Those include a VR view, i.e. a virtual view of the captured scene, composed of background 360° images and 3D objects, and a floorplan view. While the VR and floorplan views rely on offline processing by AI services only, the mobile application is expected to also include on-device AI tools for enabling DR.

The **Data** layer represents all data stores used by the ATLANTIS system, which includes a NoSQL database, storing relevant metadata of the system, as well as links to assets (e.g., images, point clouds, 3D models) stored on cloud storage (hosted on a private or public cloud server). Data stores can be accessed both from the applications as well as the services.

The **Middleware** layer serves to separate the services from the specific data structures in the data layer and applications. The components in the middleware will feed the services with the required data, and feed their results back into the data layer, while also handling the inter-relation between these different scene representations (images, geometry, scene-graphs and other metadata). They will also handle the control flow by reacting to triggers based on data being added or modified and turning them into requests to the services.

The **Services** layer represents the AI tools for non-real time processing of data. Each of the services provides a specific functionality and is deployed as a container, either on dedicated infrastructure or on the cloud (typically on hardware supporting efficient neural network inference, such as GPUs). This facilitates scaling these computation-intensive components based in a fine-grained manner.

Design Considerations

The architecture follows a layered approach, clearly separating the frontend applications, data handling and backend processing services. The main design principles are **distributed**, **service-oriented** and **event-driven**.

The architecture has been designed under the assumption of a fully distributed system, where not only the user applications but also the data stores and components may be stored at different locations and/or be provided by different organisations. This may apply to AI services that need to run on public cloud services to enable high scalability, object catalogues provided by B2B customers or user data stored with a B2B customer. With this assumption, all components will be developed by implementing the necessary precautions such as asynchronous communication and handling of failed connections in mind. Further, this architecture decouples the user-facing application layer from the functionality implementing (AI service) layer, allowing for the facile integration of new and emerging AR/VR devices (e.g. the Microsoft HoloLens 2.0), which is important in the quickly evolving landscape of interactive technologies.

Most AI and processing functionalities are provided as services, each of them performing one clearly specified function on a single or multiple data item(s), the latter focusing both on multi-modal inputs and multi-scan fusion. The services are provided as REST services with a well-defined API as well as input/output metadata structures. This allows flexibility to create pipelines of processing tasks for new data items. The services are deployed in a containerised environment, which allows for easy scalability using established frameworks and tools. The results of the services are provided back to the data layer, so that applications can access any available data, whether captured, user provided or automatically generated, in the same way. For the case of multi-modal or multi-scan inputs, these results will be associated with their corresponding data in the data layer.

The ATLANTIS system is controlled in an event-driven paradigm, where events are generated by added or modified data items (e.g., panorama being captured) or explicit user interactions. Events are also used to trigger processing by the services and notify the availability of results. In order to decouple the services from the specific data store and event mechanism, middleware components will serve connectors, feeding the REST interface with the required inputs, and providing the results back to the data store.

More information

Visit https://atlantis-ar.eu or follow us on Twitter @AtlantisAR.