SensIT AR BIM Viewer

Versions and Authors

Date	Version	Comment	Author
2021-Apr-07	1.0	Version submitted as part of Bachelor's thesis	Ali Aziz (gusaziala@student.gu.se) and Lema Rassul (gusraslea@student.gu.se)
2021-Sep-06	1.1	Refined and made compatible with treqs	Eric Knauss (eric.knauss@cse.gu.se)
2021-Oct-06	1.1.1	Refined to have consistent context diagram, use case diagram, user stories, and quality requirements. Added tracing in treqs.	Eric Knauss (eric.knauss@cse.gu.se)
2021-Oct-06	1.1.2	Refined to use template, added explanatory texts, added first functional requirements. This is ready for R1 (for which the traceability would not have been expected)	Eric Knauss (eric.knauss@cse.gu.se)
2021-Nov-12	1.1.2	Testing, if pipeline still works	

Note: This document follows the Requirements Document template as promoted in the Advanced Requirements Engineering course at Chalmers and University of Gothenburg (DAT231/DIT285). We provide some guidelines on which content to provide. Yet, you are in many cases welcome to explore alternative ways to provide the appropriate content. If in doubt, discuss with your supervisor.

> We are using the treqs open source tool to manage traceability in this document. Treqs relies on xml tags to identify requirements. These are hidden in the markdown preview and generated versions of this document, but can easily be seen in the editor. By using treqs commands such as `treqs check` or `treqs list --inlinks` these tags can be used to enable traceability and consistency checking.
>
> In the Advanced Requirements Engineering course, the requirements document is created in three increments: R1, R2, and R3. In this document, we present the requirements in relation to R1 expectations.

1. Description

> This part of the requirments document scopes the project and provides the key ideas. It is hard to imagine any project that would not benefit from making explicit the requirements on this level.

This document describes requirements for the SensIT AR BIM App, which aims to support the inspection of constructions (such as bridges). The SensIT AR BIM App should integrate with the SensIT Bim viewer, a web based app that allows to manage building information models (BIM) together with sensor data generated from sensors that are embedded in the constructs. In particular, SensIT aims to use such sensor data to predict damages, such as cracks. While the SensIT BIM viewer supports reasoning about the general state of a set of constructs (e.g. bridges), the SensIT AR BIM App supports on premise, in situ work, e.g. when assessing the quality. It could guide an assessor to the hotspots of a bridge, where based on the SensIT machine learning a crack could be most likely observed.

1.1 Goal and Scope

> This section describes the overall goal of the software. It includes a context diagram to show and define the scope.

The goal of the software is to visualize the prediction of any cracks or defects in the structure by connecting mobile, AR enabled devices with the SensIT BIM viewer infrastructure. The context diagram in Figure 1 shows the AR viewer in its context as well as the interfaces to users and other systems that it must support.

> PlantUML code shown only for demonstration. Ideally, the code would be commented out and the UML diagram it generates would be shown. PlantUML does not directly support Context Diagrams. There are workarounds, but the result in more complicated code.

```
@startuml sensit-context

actor client
actor contractor
```

```
  component bim as "BIM Database"
  component sens as "Sensor Data"
  component ml as "ML Predictions"

storage "Inspection of construction" #palegreen;line.dashed{
  component ar as "AR Viewer"


  inspector <<->> ar :(e1) view, (e2) find hotspots, (e3) annotate
  bim <<-->> contractor:maintain
  client <<->> ar :(e1) view, (e3) annotate
  client <<->> bim:maintain

  ar <<==>> bim:(e4) overlay, \nadd annotation
  ar <<== sens
  ar <<== ml

}
@enduml
```

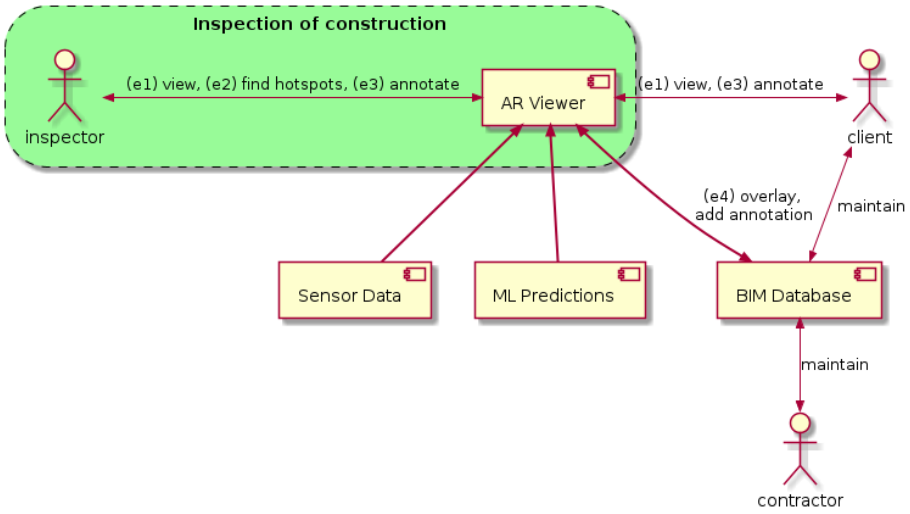

Figure 1: sensit-ar-usecases

Figure 1. Context diagram of AR Viewer.

Legend:
- We use storage with green background and dashed line to show the domain
- We use arrows to show interaction and the text to denote the key events

Withing this scope, SensIT AR aims to reach the following business goals:

1.1.1 Phases

Phases of construction projects are provided as additional information to help readers that are not familiar with the domain. These are not requirements for the AR viewer, just background information.

The SensIT AR BIM App is embedded in the following high-level process of modern construction.

1. Design, Client and Contractor agree on process and design of the construction. Sensors and models are decided on.
2. Production, Contractor produces the construction. AR can help to monitor the production.
3. Warranty phase, Contractor still responsible for maintenance. AR can help to assess the constructions and identify potential problems.

1.2 Business case and stakeholder map

This sections describes who will be using the software, which other stakeholders exist, and how the software will be generally worth its development

1.2.1 Business Goals Within the scope depicted in Figure 1 and in relation to the phases, SensIT AR aims to reach the following business goals:

Business Goal 1: Increase effectivity of inspection. SensIT AR BIM Viewer should increase the likelihood of detecting potential problems (i.e. those that should be reported) during inspection. This is done by providing an "inside view" of the construction, e.g. of a concret beam based on sensor data and machine learning based prediction, which is shown as a heatmap overlay in the AR.

Business Goal 2: Increase quality of inspection report. SensIT AR BIM Viewer should increase the quality of the report by providing new means of locating findings in relation to the construction through AR and by synchronizing this information with the web based bim viewer.

In the following sections, we will define the events and quality attributes in more detail. The Table 1 shows how these events and quality attributes relate to the business goal:

1.2.2 Goal Domain Tracing

Goal domain tracing allows to relate goals to events and qualities. These will be detailed below.

Table 1. Goal domain tracing

Goal	(e1) view	(e2) find hotspots	(e3) annotate	(e4) overlay	(qr1) accuracy
Business Goal 1.	x	x	x	x	x

Goal	(e1) view	(e2) find hotspots	(e3) annotate	(e4) overlay	(qr1) accuracy
Better inspection. **Business Goal 2.** Better inspection report.			x	x	x

1.2.3 Stakeholders SensIT AR has three high-level stakeholder groups (see Table 2).

Table 2. Stakeholder Map for SensIT AR.

Name	Relationship	Representative	Power
Client	Potential Customers	Trafikverket	High, may fund product
Inspector	Potential Users	WSP	High, main users of the product
Contractor	Potential Users	NCC	High, another user of product

Stakeholder 1: Inspector. The inspector is tasked with the inspection of a construction site. This can be a consultancy company hired for this taks. In Sensit, the Inspector is represented by WSP.

Stakeholder 2: Contractor. The contractors are the ones in control of the production phase of the constructions. The contractors want access to the BIM viewer during the inspection and control phase of the project. An example of what the contractor wants visualized in the BIM viewer is the temperature during the casting of the structure. In Sensit, the Constructor is represented by NCC.

Stakeholder 3: Client. The client wants to see that their properties are working efficiently. If the client decides to order an inspection they must hire an inspector. After the client receives the construction contract from the contractor, they may also receive the temperature view as a description of the delivered state. This would be a change in the current way of working introduced by Sensit. In Sensit, the Client is represented by Trafikverket.

1.3 Core Functionality

This section describes the core functions that the software will provide. - It includes a *use case diagram* or *goal model* that provides an overview of key actors and key functions. - IDs allow to trace between elements (use cases) of the model and more detailed descriptions of these use cases in the text.

1.3.1 Relation to SensIT Infrastructure and BIM Viewer Through the connected SensIT infrastructure, a contionus monitoring will occur as well as systematic evalution of the effects of deformation, leakage, crack development,

corrosion, limescale leaching etc. This will be conducted through the use of cloud-based machine learning and sensors that are built into the structures. The sensors will continously send the acquired data to the cloud service that will then perform calculations and visualize the results through the BIM viewer.

1.3.2 Overview of Functions in SensIT AR BIM viewer

```
@startuml sensit-ar-usecases
left to right direction
actor user
user <|- inspector
user <|- contractor
user <|- client

rectangle "SensIT AR BIM Viewer" {
  "UC1: access BIM" as (access)
  "UC2: check site" as (check)
  "UC3: follow-up" as (followup)
  "UC4: annotate structure" as (notes)

  "UC7: view structure in AR" as (view)
  "UC6: find hotspots" as (find)

  (check) ..> (find) : <<include>>
  (followup) ..> (find) : <<include>>

  user -- (access)
  user -- (check)
  user -- (followup)
  user -- (notes)

  (view) ..> (overlay) : <<include>>
  (view) <. (find) : <<include>>
  (overlay) <. (...with BIM) : <<extends>>
  (overlay) <. (...with sensor data) : <<extends>>
  (overlay) <. (...with heatmap from ML) : <<extends>>
  (overlay) <. (...with annotations) : <<extends>>

  (notes) <.. (upload picture) : <<extends>>
}
(overlay) -- :BIM database:
(access) -- :BIM database:
(notes) -- :BIM database:

"UC5: make structure/BIM available" as (share)
contractor -- (share)
```

```
(share) -- :BIM database:

note "Machine learning \n results are computed \n offline and stored \n in BIM database." as

note top of user
  User must be
  authenticated
end note

(...with heatmap from ML) .. N

@enduml
```

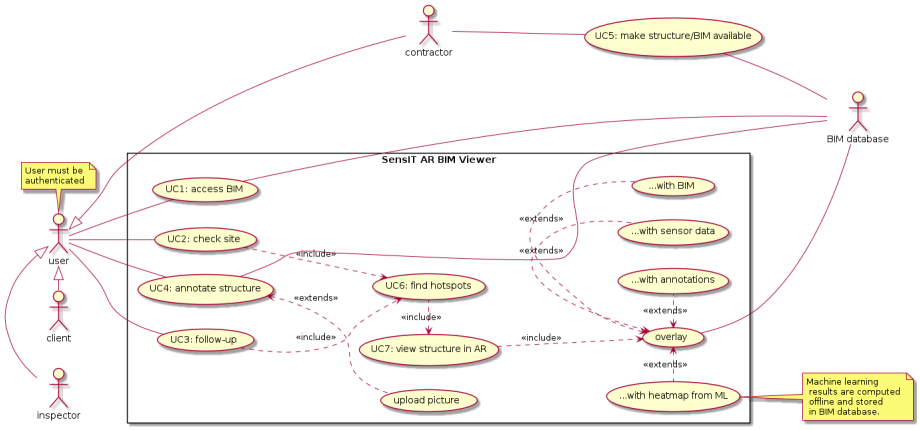

Figure 2: sensit-ar-usecases

Figure 2. Use case diagram with core functionality of the AR Viewer.

The main use cases for the AR visualisation tool are presented in the use case diagram in Figure 2. The SensIT AR BIM Viewer is set not only to visualize areas of interest to inspectors during inspections, but also allows the user to review sensor data displayed in plots both before and during the inspection. The inspector is then also able to create annotations during the insepction to review the location and comments when returning back to the office. Previous notes from other inspectors can also be reviewed to make an assessment of any changes or causes for any discovered damage.

> The use cases from Figure 2 are elaborated by using the user story format: `As a USER I want FEATURE so that BENEFIT`. This allows to connect the benefit to the business goals above.

The following user stories elaborate on these main use cases and describe how value to each stakeholder is provided:

UC1: As a user, I want to access the BIM remotely and at all times so that I can for example check the BIM while on site.

UC1.1: As a user, I want to select the construction site in the BIM Viewer.

UC1.2: As a maintainer, I want to see potential cracks in the BIM of a particular construction.

UC2: As an inspector, I want to check a construction site for potential problems

UC3: As an inspector, I want to follow-up on previous defects within the construction so that the development of deflections, cracks, or strains can be monitored.

UC4: As an inspector, I want to annotate a finding on a structure, so that potential defects such as deflections, cracks, or strains can be followed-up on. A picture may be uploaded to show the finding. The AR viewer should properly locate the annotation on the virtual structure.

UC5: As a maintainer, I want to share the construction site in the BIM Viewer with collaborators.

UC5.1: As a maintainer, I want to be able to authorize users to access the application

UC6: As a user I want to find hotspots so that I can focus my inspection of a construct.

UC7: As a user I want to view a structure in AR so that an overlay from the BIM database is provided on site.

1.4 Performance Requirements, Specific Quality Requirements, Constraints

> This section describe the most important non-functional requirements in relation to the project goals.

The following table ("Quality Grid") lists relevant quality factors and their estimated importance for the success of the project.

Quality factors for SensIT AR BIM Viewer	Critical	Important	As usual	Unimportant	Ignore
Operation					
Integrity/Security			x		

Quality factors for SensIT AR BIM Viewer	Critical	Important	As usual	Unimportant	Ignore
Correctness/Accuracy	QR1				
Reliability/Availability			x		
Usability		QR2			
Efficiency/Performance		QR2			
Revision					
Maintainability			x		
Testability			x		
Flexibility			x		
Transition					
Portability					
Interoperability			x		
Reusability			x		
Installability			x		

A few qualities appear to be particularly important:

QR1: Accuracy.

Accuracy of AR overlay and location of potential problems is crucial for the SensIT AR BIM Viewer. Without sufficient accuracy, the AR viewer will not be useful.

Examples of Accuracy: - Example 1: If a potential problem is shown, it must be accurately placed. - Example 2: The alignment of the AR overlay (heatmap, wireframe) with the real world structure must be exact.

Open question: We also must find ways to encourage inspectors to look beyond the displayed potential problems.

QR2: Performance.

The AR interface must load relevant information sufficiently fast so that it does

not affect usability.

Note on Performance: Performance is secondary, but it should not affect usability: Alignment should be quick and accurate enough, so that the app is useful.

2. User Requirements Specification

In this part, more detailed requirements are provided from the perspective of users. The goal is to provide problem-based requirements, i.e. to not foreclude any design decisions.

Most projects in real life do benefit from a concise description of this content. Agile projects in particular spread out this information or leave it for the agile team to discover them during the sprint. A large agile project would be well advised to document such content explicitly. Our company partners have started to manage this content as part of the source code repository, together with software and tests. Then, it is the responsibility of the agile teams to maintain the information.

It can be good to re-organize this section in a different way. - Especially for complex products, services, projects, you can consider to split it down by workarea or component. Then, the following headings are one level deeper, under each component.
- It can be good to change the order, e.g. start with functional requirements, then data, etc. - If in doubt, discuss with your supervisor

2.1 Data requirements

Provide a class or ER (or: class) diagram, a data dictionary, and requirements as needed. Keep them on a high level though, so that you can rely on these data items in your description of functional requirements. Make sure to explain all diagrams in text, e.g. through data dictionary. If you are tempted to add more detail to the data, move those details to Section 3.3 instead.

Data requirements should be available from R2 on.

2.1.1 Class diagram **DR1:** SensIT AR BIM Viewer shall store and process according to the following class diagram.

```
@startuml sensit-ar-classes
class ARViewer {}
package BIMDatabase {
    class BIMProvider {
        BIM getBIM(String name)
    }
    class BIM {
```

```
        String name;
        String description;
    }
    class Wireframe {
        Point.3D[][] wires;
        Referencepoint
    }
    BIM *- Wireframe
}
package SensorData {
    class SensorDataProvider {
        SensorData getSensorData(String name)
    }
    class SensorData {
        String name;
        String description;
        Point.3D[] point;
        Number[] value;
    }
}
package MLPredictions {
    class Predictor {
        SensorData[] predict(SensorData[] input)
    }
    class TextureGen {
        Texture[] generateHeatmap(BIM bim, SensorData[] input)
    }
    class Texture {}
}
ARViewer o-- BIMProvider
ARViewer o-- SensorDataProvider
ARViewer o- Predictor
ARViewer o- TextureGen
@enduml
```

2.1.2 Data dictionary

Class: BIM

The building information model stores information about a structure in a defined format. We simplify this information for the purpose of this specification. Beyond a name and description, we are mainly interested in the wireframe that represents the structure in augmented reality.

Examples

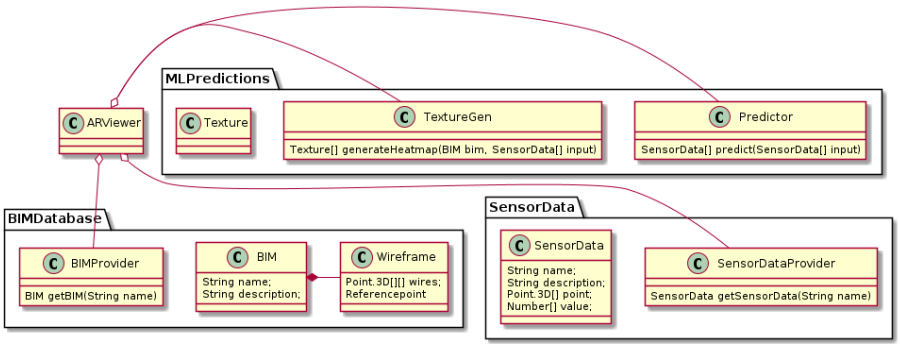

Figure 3: sensit-ar-classes

1. A bridge
2. A building
3. An element (e.g. a column) of a bridge

Attributes

name The name of the building. Must be unique.
decription additional text to describe the building

Class: BIMProvider

The BIMProvider allows the ARViewer to select BIMs by name for presenting them in augmented reality.

Methods

getBIM(String name):BIM this method retrieves a specific BIM from the BIM-Database and makes it available to clients such as the ARViewer.

Class: SensorData

This class represents data from a particular sensor. Sensor values are related to a point in 3D space, relative to the reference point in the wireframe of a BIM. Each point has an individual value attached to it.

Examples:

1. Fiber based sensors that measure distortion of structures

Attributes

name The descriptive and unique name for this sensor
description a more verbose description that can be shown to users as help text
points an arry of 3D points relative to the referencepoint of a wireframe
values an array of numbers that represent the sensor value at each of the points

---

Class: TextureGen

A helper class that generates textures for wireframes based on sensor data.

Examples:

1. Generate a heatmap to be shown on the top of the wireframe to show how much distortion was measured in which location.
2. Show the steel fortification within the concrete structure

Methods:

generateHeatmap(BIM bim, SensorData[] input) : Texture[] Generates a heatmap to be shown as an overlay over the wireframe of the BIM

Class: Predictor

A class that derives predictions based on SensorData. A predictor is typically AI-based.

Examples:

1. Predict likelyhood of cracks appearing in certain areas

Methods:

predict(SensorData[] input) : SensorData[] Represent the predictions as sensordata, so that they can easily be shown based on TextureGen.

2.2 Functional requirements

Provide a list of features or task descriptions that give detail to the core functionality in Part 1 (High-Level Description). Make sure that each functional requirement relates to business value and its importance to particular stakeholders as well as project success. You could consider user stories here, but the section should provide more detail than Section 1.3. Perhaps use the user stories in Section 1.3 as the Task Name of your task descriptions and then add the

additional information for a task description, task and support, or even screens and prototypes (but then, perhaps start working on your UI Prototype in Section 3.2 and rather provide strong tracing to the screens in that prototype instead to these task descriptions). Rely on L4 content to shape the content here.

An initial draft of functional requirements should be available in R1. Detailed requirements should be available from R2 and refinements should be provided in R3 of this document.

FR1: The system shall allow to access the BIM by supporting Task 1.

Task 1	Access the BIM	
Purpose:	Support accessing the BIM from a mobile, AR enabled device. Restrict access to authorized users.	
Trigger:		
Pre-condition:	User is authorised to access BIM and authenticated.	
Frequency:	Users might access the BIM in their daily work.	
	Sub-tasks:	**Example Solution:**
1.	Select BIM	Show a list of BIMs that the user is authenticated to see
2.	Authenticate user	BankID, biometrics, pin code
3.	Show BIM	Show on display of mobile device. Consider using VR capabilities to show BIM.
	Variants:	
1a.	If the correct BIM cannot be selected, allow user to *request authoritization**	
2a.	If user cannot be authenticated, ask to *create user* or *recover access*	

Task 1	Access the BIM	
2b.	If user is not authorized, allow user to *request authoritizatio*	Allow adding freetext to describe why access to this BIM is needed.

FR2: The system shall allow to check a construction site by supporting Task 2.

Task 2	Check construction site	
Purpose:	Inspection of a construct	
Trigger:	Inspection of a construct is to be started.	
Pre-condition:	Inspector is on site and has AR enabled mobile device. Inspector has accessed BIM as in Task 1.	
Frequency:	Inspection is triggered in regular intervals, depending on the type of construction.	
	Sub-tasks:	**Example Solution:**
1.	General manual inspection of construct.	Usually includes visual inspection as well as testing structure, e.g. with a hammer. Might ask inspector to scan certain QR codes to show that they have visited area.
2.	*Find hotspots (Task 6)* for additional focused inspection.	Idea is to use AR overlays to guide inspector. Might ask inspector to scan certain QR codes to show that they have visited area.

Task 2	Check construction site	
3.	*View structure in AR (Task 7)* for improved visual inspection.	Idea is to use AR to overlay structure, e.g. showing the steel fortifications within the concrete, the sensor data, previous annotations and pictures, and predictions of defects based on machine learning.
	Variants:	
1a.	A follow-up (as in UC3) is a variant of this task, where subtask 1 may be skipped.	

FR3: The system shall allow to annotate structures by supporting Task 3.

Task 3	Annotate structure	
Purpose:	Support following up on potential defects such as deflections, cracks, or strains.	
Trigger:	Potential defect found during checkign a construction site	
Pre-condition:	Inspector is on site and has AR enabled mobile device. Inspector has accessed BIM as in Task 1.	
Frequency:	Might be triggered severel times during checking a site.	
	Sub-tasks:	**Example Solution:**
1.	Locate potential defect in AR	

Task 3	Annotate structure	
2.	Add annotation with basic information **Variants:**	perhaps a form or audio recording
2a.	Take a picture and upload	

Template for task descriptions | Task NR | Task name | | | —: | — | — | | Purpose: | | | | Trigger: | How triggered | | Pre-condition: | What must be in place |
| Frequency: | How often | | | **Sub-tasks:** | **Example Solution:** | |
1. | a subtask | a solution idea | | | **Variants:** | | | 1a. | a variant | |

2.3 Detailed Performance Requirements, Specific Quality Requirements, Constraints

Based on your prioritization in Section 1.4, Use *PLanguage* or similar to describe (only) the important quality attributes (or: non-functional requirements) in detail. Depending on your project and the rest of this document, there are a lot of good options: - Consider tracing to or from functional requirements or data requirements to describe quality in context. - Consider moving some detailed quality requirements to a new Section in Section 3, especially if they relate to internal qualities directly connected to the system.

Detailed non-functional requirements should be provided from R2 on.

QR1.1 Accuracy of AR Overlay [TAG]	How accurate the AR overlay matches the real structure
SCALE:	Likert scale for test users, does the accuracy of the AR overlay affect the usefulness of the app (strong disagree, disagree, agree, strong agree)
METER:	Measured during acceptance test with 5 Inspectors
MUST:	Bias towards disagree
WISH:	Mainly strong disagree
PLAN:	_____ (supplier, please specify)
PAST:	No AR overlay available in the past

QR1.2 Accuracy of locating annotations and potential problems [TAG]	How accurate potential problems and previous annotations are shown in the AR interface on the real structure
SCALE:	Offset in any direction in cm
METER:	Measured during acceptance test, estimated by testers.
MUST:	No more than 50cm
WISH:	No more than 5cm
PLAN:	_____ (supplier, please specify)
PAST:	Location described in inspection journal, usually able to find the spot within a search field of 100cm to 300cm.

2.4 Proposed prioritization

> Make explicit how functional requirements were prioritized. Use the subsections here to provide guidance to the supplier about priorities. Do not hestitate to also reveal here the analysis (e.g. $100 method) and its results.
>
> Prioritizations should be provided from R2 on.

Prioritization is based on a workshop with representatives from all stakeholder groups. Accuracy is the most important quality requirement and technical risk. In order to mitigate this risk, the first task that needs to be supported (e.g. with dummy data) is Task 2.

2.4.1 Next release

> Describe which requirements to cover in the next release of the system under construction (i.e. the one you defined in your project mission and that you are writing requirements for, NOT R1, R2, R3 in this course).

The first release of SensIT AR BIM Viewer should support Task 2 and demonstrate acceptable accuracy as defined in QR1, QR1.1, and QR1.2.

2.4.2 Second release

> Describe which requirements to cover in the second release. This is a best practice, if you ever have to schedule tasks: Always also plan for the iteration/release/increment after the one that comes next. It makes it easier to focus on the most important things in your next iteration.

The second release of SensIT AR BIM Viewer should fully integrate with the existing BIM database and support Task 1. This will allow to test whether the AR viewer has a positive impact on inspections.

2.4.3 Future releases (optional)

> Describe which requirements not to cover in the first two releases. Note, that these could still be important requirements, but perhaps, based on interdependencies and relation to business goals / priorities of certain stakeholder groups, they might not be urgent. Make sure that a reader can understand your decisions.

Task 3 should follow last and allow to capture any issues found during inspection within SensIT AR BIM viewer and the BIM database.

3. System Requirements

> In this part, more detail is provided and requirements are specified from the perspective of the system. Requirements are much closer to specific solution ideas or design decisions.
>
> Modern agile projects may rely on the actual artifact instead, e.g. early version of the software itself instead of prototypes, the data base schema instead of data requirements, and automated tests of some sort. Such projects often suffer from a lack of overview and fail to properly include UX experts in the development or to maintain their data schemas or effective test suites. However, in a typical, fast-paced agile project, this content may get outdated too quickly. Especially low level functional requirements then become cumbersome to maintain. For a supplier, this level of requirements can be crucial to maintain: How are the customer requirements thought to be covered?
>
> For the scope of this course, it is our goal to show some awareness for this perspective, especially to avoid unrealistic requirements in Part 1 and 2. You may focus and provide such detailed requirements only for certain parts.
>
> Content in Section 3 could be available as draft in R2 and must be included in R3.

3.1 System requirements

> Detailed requirements from the perspective of a system. This could be feature requirements ("The system shall…"), consider EARS notation (https://alistairmavin.com/ears/) If no more detailed requirements can be reasonably added, consider providing acceptance tests instead (Section 3.4)

3.2 UI Prototype

Sketch the most important UI elements of the project, e.g. screens. Arrange them to show how core functionality and critical attributes are supported. An idea that might work for you is to rely on an activity diagram syntax and replace the activities with prototype screens.

3.3 Detailed Data Requirements

Specify constraints and detailed properties of data as needed.

3.4 Acceptance Tests (Optional)

Describe how to determine whether the product is acceptable with respect to core functionality and critical attributes. This is not a focus within this course, but should be a concern in a real world project. This can be a useful alternative, if no more detailed requirements can be reasonably added in Section 3.1