CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF TRANSPORTATION SCIENCES

LINKÖPING UNIVERSITY

FACULTY OF SCIENCE AND ENGINEERING

Bc. Jan Štůla

DESIGN AND DEVELOPMENT OF ANS HELP DESK WITH ASSET MANAGEMENT

Master thesis

2021

CZECH TECHNICAL UNIVERSITY IN PRAGUE
Faculty of Transportation Sciences
Dean's office
Konviktská 20, 110 00 Prague 1, Czech Republic

K614 .... **Department of Applied Informatics in Transportation**

MASTER'S THESIS ASSIGNMENT

(PROJECT, WORK OF ART)

Student's name and surname (including degrees):

Bc. Jan Štůla

Code of study programme code and study field of the student:

N 3710 – IS – Intelligent Transport Systems

Theme title (in Czech):	**Asset Mangement a Design a Tvorba HelpDesku pro ŘLP**
Theme title (in English):	Asset Management and Design and Development of ANS HelpDesk

Guides for elaboration

During the elaboration of the master's thesis follow the outline below:

- Air Navigation Services of Czech Republic
- HelpDesk
- Legislative aspects in the project
- Asset Management
- Tools for the Design and Development
- Design of the HelpDesk
- Development of the HelpDesk

Graphical work range:	Determined by the supervisor
Accompanying report length:	55
Bibliography:	ISO/IEC 19505-1:2012(E) Axelos. (2019). ITIL Foundation : ITIL 4 Edition. The Stationery Office. ISO/IEC 27000 series Czech law number 181/2014 Col.

Master's thesis supervisor: **Ing. Ota Hajzler**

Tatiana Polishchuk PhD

Ing. Karel Pohl

Date of master's thesis assignment: **August 16, 2020**
(date of the first assignment of this work, that has be minimum of 10 months before the deadline of the theses submission based on the standard duration of the study)

Date of master's thesis submission: **August 9, 2021**
a) date of first anticipated submission of the thesis based on the standard study duration and the recommended study time schedule
b) in case of postponing the submission of the thesis, next submission date results from the recommended time schedule

L. S.

....	....
doc. Ing. Vít Fábera, Ph.D. head of the Department of Applied Informatics in Transportation	doc. Ing. Pavel Hrubeš, Ph.D. dean of the faculty

I confirm assumption of master's thesis assignment.

....
Bc. Jan Štůla
Student's name and signature

Prague ....August 16, 2020

Acknowledgement

I would like to thank Czech Technical University in Prague as well as Linköping University for the opportunity to study the double degree master’s programme. I would like to thank my supervisors, Ing. Ota Hajzler, Tatiana Polishchuk, PhD. I would also like to thank Ing. Karel Pohl from Air Navigation Services of the Czech Republic for guidance, helpful critique, and the time he invested in me. Finally, I would like to thank my family and my life partner for their moral and material support during the creation of the thesis and the studies.

Declaration

I hereby declare that the presented master thesis is my work and that I have cited all sources of information following the Guideline for adhering to ethical principles when elaborating a final academic thesis. I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Goll., the Copyright Act, as amended that the Czech Technical University in Prague has the right to conclude a license agreement on the utilisation of this thesis as schoolwork under the provisions of Article 60(1) of the Act.

In Prague 9th of August 2021 ……………………………………………………

Signature

CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF TRANSPORTATION SCIENCES

LINKÖPING UNIVERSITY

FACULTY OF SCIENCE AND ENGINEERING

DESIGN AND DEVELOPMENT OF ANS HELP DESK WITH ASSET MANAGEMENT

Master thesis

August 2021

Bc. Jan Štůla

ABSTRACT

The subject of this master thesis is to prepare specifications, project management strategy, conduct IT business analysis, and model the help desk system for Air Navigation Services of the Czech Republic. The interviews with the customer created a base for the specification, the analysis, and the model. The project uses ITIL for design, Agile PRINCE2 for project management, and UML and ERD for modelling.

KEY WORDS

Help desk, Service desk, IT asset management, Air Navigation Services of the Czech Republic, Project management, Design, Model-based development, Agile, Waterfall, Unified modelling language, UML, Entity-Relationship Diagram, ERD, Requirement gathering, Market research, SWOT analysis, Risk management, Time plan, Gantt chart, Improvement Management, ITIL

Table of Contents

List of Abbreviations

Abbreviation	Meaning
AFM	Alstanet Facility Management
ANS	Air Navigation Services
ANS CR	Air Navigation Services of the Czech Republic
API	Application Programming Interface
ATC	Air Traffic Control
ATCO	Air Traffic Control Officer
ATM	Air Traffic Management
BMC	American software company – Boulette Moores Cloer
CI	Configuration Item
CIR	Continual Improvement Register
CSF	Critical Success Factor
DevOps	Development and Operations
DTECH	Technological department
EA	Enterprise Architect
ERD	Entity-Relationship Diagram
EUROCONTROL	European Organisation for the Safety of Air Navigation
FAQ	Frequently Asked Questions
FK	Foreign Key
FOO	Flight Operations Officer
FTA	Fault Tree Analysis
GPS	Global Possitioning System
HTML	HyperText Markup Language
ID	IDentification number
ISO	International Organization for Standardization
IT	Information Technology
ITIL	IT Infrastructure Library
ITSM	IT Service Management

Abbreviation	Meaning
IVR	Interactive Voice Response
KPI	Key Performance Indicator
MIS	Company information systems
MS SCCM	Microsoft System Center Configuration Manager
MTBF	Mean Time Between Failures
MTRS	Mean Time to Restore Service
PK	Primary Key
PKT	ANS CR, DTECH, technical unit
PRINCE2	Projects IN Controlled Environments 2nd version
PST	ANS CR, DTECH, server unit
RIIT	ANS CR, DTECH, development unit
SMART principle	specific, measurable, achivable, relevant, time-bound
SMS	Short Message Service
SQL	Structured Query Language
SWOT	Strengths, Weaknesses, Oportunities, Threats
TIS	Technical Inforamtion System (Technický Informační systém)
UCL	Civil aviation authority (Úřad pro Civilní Letectví)
UI	User Interface
UML	Unified Modelling Language
WBS	Work Breakdown Structure
XML	eXtensive Markup Language

1. Introduction

Air traffic fell hard due to the Covid-19 crisis. On the other hand, those who fly must still perform the same actions as before. The Air navigation services (ANS) of the Czech Republic are composed of two parts. The active part, which performs the air traffic control, and the administrative part. To keep the spread of Covid-19 at a minimum, most people started working from their homes. Only the air traffic controllers and few others kept on working from the ANS's buildings.

ANS is an organisation like every other. Though the state owns it, they still have customers, partners, and other stakeholders. They provide their customers with services as well as other organisations provide services for them. Every company should improve themselves, even the smallest ones. According to the information technology infrastructure library (ITIL), the first step in the improvement process is to create a vision.

Currently, there are two systems used as help desk and service desk in ANS. They are robust in a lot of ways and expensive. A single help desk application aims at all employees, and it will help ANS improve service provision and management. The vision is to create one system where the users could submit their problems and proposals for improvements. The system should be easy to use while taking a minor portion of the user's time to use. The same is true for the admin side of the system. They should be able to view the problems and easily divide the work. The system should also include asset management, in which the admins could review the equipment and programmes.

In this thesis, we have used the ITIL improvement guideline. In the first chapter, we went through the theoretical basis for the thesis. However, thanks to the references to this chapter, it is not necessary to read it first. Then, we analysed the current state of ANS and their help desks. We learned more about the future state by collecting the requirements. In the fourth chapter, we have planned out and modelled the implementation of it. Finally, we have given a proposal on how to continue with the work. That means how ANS should carry out the plan, assess the result and keep the momentum going.

2. Theoretical basis

Before we dive into the project, we will go through the theory. There are many references to this chapter in other chapters. Therefore, it is unnecessary to read all sections in this chapter now, but rather read particular parts whenever there is a reference. On the other hand, it is also good to read this part too.

First, we will look at the Information technology infrastructure library (ITIL). We will use it to describe Air navigation services of the Czech Republic (ANS CR) and understand the needed services for the project. We will also use it a bit for project management. Then we will look at some project management techniques and requirement gathering. Finally, we will look at the graphical modelling languages because they are essential for model-based development.

2.1. Information technology infrastructure library

ITIL was created in the 1980s in Britain based on ISO 20000 (see, e.g. [2]). It is a set of best practices for information technology (IT) service management. It fast became used by major companies all over the world. It is suitable for creating an overview of the organisation, assess its weaknesses and strengths, and the process of creating value for them and others. ITIL is described in five essential element guides, each containing one of the significant areas of service management: service strategy, service design, service transition, service operation, and continual service improvement. Besides the books, training opportunities are ending with a certification. Axelos is a company that governs the certification and the evolution of ITIL. The certificates are of three types depending on the depth of the knowledge. [1]

This section is not for explaining ITIL in depth. In this section, we will go through the basis of ITIL for the practical part of the thesis. We will be using ITIL to describe Air Navigation Services (ANS) of the Czech Republic in another section. The description is suitable for the development of the help desk and any other future system. However, we first need to define the key concepts and learn more about ITIL. We will be using Axelos's ITIL Foundation for ITIL 4 (see [4]) and some training materials (see, e.g. [3], [10]).

In the following section, we shall dive into improvement management. That might seem weird for those who know ITIL. On the other hand, this thesis is a part of an improvement project. If we describe ITIL as described in [4], we might dry up and lose consistency. After the improvement management, we will go through the organisations and describe them using ITIL. Last, we will look at the management practices, which might be relevant to the thesis somehow.

Before we do so, we should elaborate on few terms. Table 1 is a list of terms and their explanations. There are much more terms in ITIL, but we will not need them for the following sections.

Table 1: Some ITIL terms explained

Term	Explanation
Organisation	An organization is a person or a group of people. There might be inner organisations, and one person might be in multiple organisations.
Value	For example, money, experience, connections, or resources. Anything of value for an organisation.
Product	A think which an organisation creates using their technology, information, suppliers, partners, and people.
Service	A set of products. The organisation continuously provides it.
Service Provider	The organization/s which provide the service
Service Consumer	The organization/s which receive the service
Service value system	A service value system is a model of creating value from opportunity and demand concerning the organisation, its culture, processes, or practices.

2.1.1. Improvement management

The organisation should always search for ways how to improve itself at all levels and in all areas. We can improve the service value system, as well as the outputs, inputs, or relationships. The ITIL service value system includes how the organisations should approach the implementation, improvements themselves, and continual improvement. Figure 1 shows the continual improvement model and which activities an organisation takes to improve. [4]

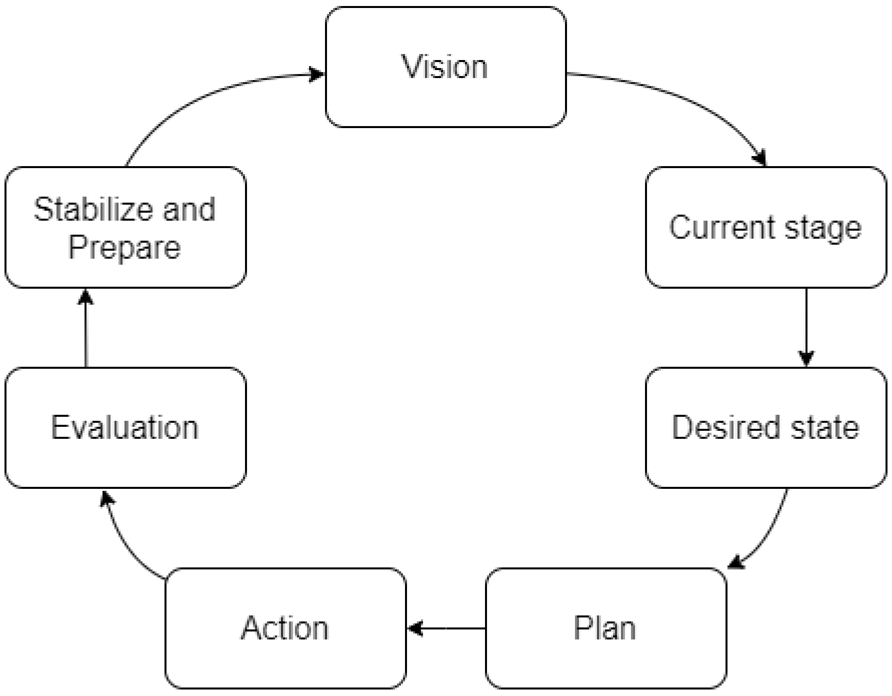

Figure 1: ITIL's continual improvement stages

First, the organisation needs a vision for improvement. Anyone inside or outside the organisation might come with one. The vision should give context for all subsequent decisions and link the actions. Naturally, the vision should support the whole organisation's goals and objectives. [4]

Next, the organisation should asset their current state. The organisation should analyse their existing services and how the users perceive the value in them. We should also analyse the organisation itself. That means the people, their competencies and skills, technologies, capabilities, and processes and procedures. We must consider the organisation's culture along with the politics. A good understanding of the current state of the organisation leads to appropriate and successful improvements. We can use strengths, weakness, opportunity, and threat (SWOT) analysis (see Section 2.3.1), a balanced scorecard review, internal and external assessments and audits, or perhaps even a combination of several techniques to analyse the current state better. [4]

The vision defines the future state, which the organization cannot achieve after a single improvement. The organization must divide the route to the vision into pieces. A single step is an improvement from a current state to the next one. We use the description to understand better which parts of the organisations will change. In this stage, we define objectives, critical success factors (CSFs) and key performance indicators (KPIs). While creating those, we should follow the SMART principle. That means that the objectives, CSFs and KPIs should be specific, measurable, achievable, relevant, and time-bound. [4]

After defining the starting, following, and final point, we need to plan to move to the next step. Generally, there is not a single solution for the improvement, and we might have to assess which is the best approach. We can divide improvement into multiple steps, similarly as we did with the vision. In this case, plan the improvement to evaluate the progress and eventually change the plan after each iteration. [4]

Independent of the size of the improvement and its impact, we should continually update the plan while executing it. That ensures flexibility and helps the risk management, change management, measurement and reporting and continual improvement. On the other hand, it is also essential to keep the objectives in mind. [4]

Once we set the plan in motion and the organisation continues the improvement, they need to observe the actual changes. The organisation needs to evaluate whether the changes and actions were successful. The organisation also needs to check the relevance of the improvement and its objectives. We need to do the checks during the actions and after them and eventually update the plan. [4]

After the organisation finishes an improvement, they might go right to another. On the other hand, the organization must not forget improvement. If the improvement was successful and brought the expected value, the marketing should take over and emphasise it. That prepares the organisation for the next improvement and initiates it. [4]

2.1.2. Modelling organisations and their services

Organisations compose out of one or more people or organisations. In ITIL, the organisations provide services and sell products to create value. The value can be material or imaginary. Generally, someone gets money, which naturally is valued too. On the other hand, organisations can, for example, obtain experiences, improve their processes, create new relationships, or gain new employees. Organisations generally both provide and are provided with services. From a more macroscopical view, the organisations create a chain (see Figure 2). [4]

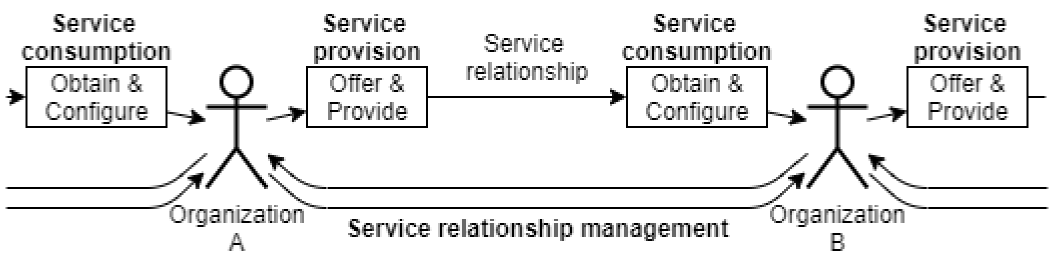

Figure 2: Service relationship

Microscopically, each organisation has its service value system (see Figure 3). Every organisation has partners, suppliers, consumers, inner organisations, and other stakeholders from a mesoscopic view. There are services between those organisations. ITIL also provides guiding principles and improvement management. It also provides practices from general, service, and technical management. To read more about ITIL see e.g. [1], [3], or [4].

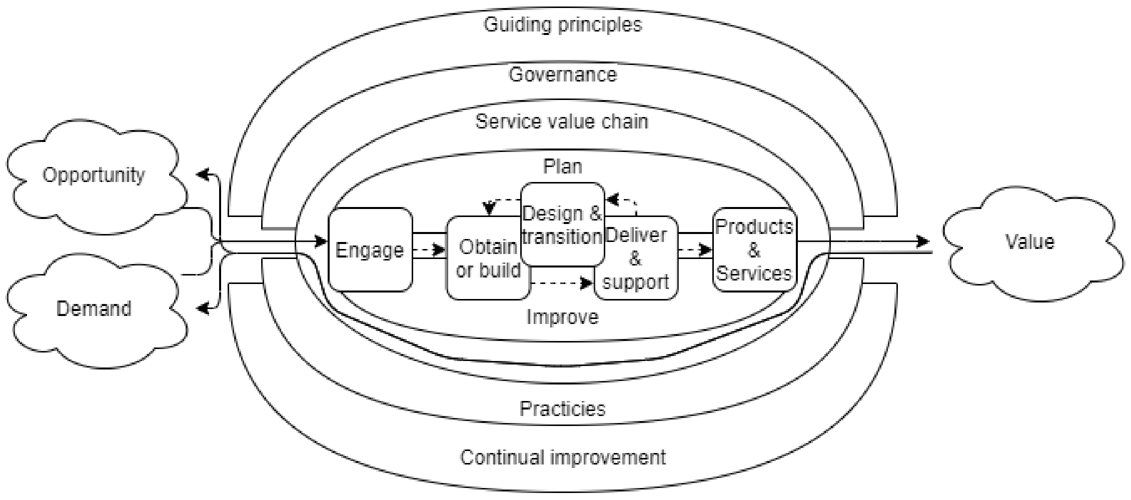

Figure 3: Service value system

An accurate description of an organisation contains suppliers and partners, inner organisations, service consumers for the organisation and other stakeholders. It must also contain the services. A detailed description then might contain how the organization conducts each service. That means the description of the service value system for each service. This description is relevant the most for business and management.

2.1.3. Key management practices

In this section, we will look at some management practices from [4]. Management practices are practices which the organisations should, and most of the time already did adopt. There are three groups of practices; general, service, and technical. We are interested in practices among these groups, but not all practices are relevant for us. To read more about them, see [4]. We will fully describe the practices we will use to manage the project and design the system. We will go through those practices that ANS CR might use during the project's development, implementation, or evaluation. We will also look at those practices, which define this project. Finally, we will mention those, which might inspire the further improvement of the system.

First, we will look at a practice, which the project is part of. **Continual improvement** is a general management practice. Its goal is to encourage improvement, identify improvement opportunities and lead the improvement projects according to the improvement management. It is possible to divide an improvement into a multi-phase project or keep it as a single one.

This project is a part of continual improvement, and the application will help ANS CR in future improvement projects. We will not look at the improvement management closer because we described it already in Section 2.1.1. There are three approaches to continual improvement; lean, agile, and DevOps. Lean methods try to eliminate wastes, the agile method looks only to the future, and DevOps improves improvements. Organisations should prefer one method but not exclude others. Employees can also improve themselves. There should be an internal organisation that focuses on the improvement of employees. On the other hand, it is possible to outsource this service. The management should store data from the improvements to navigate through organisation strengths and skills. The organisations store the data in databases or documents called continual improvement registers (CIRs). The organisation can, and should, collect the data at any opportunity, for example, during project development, morning coffee, or contract negotiations. A continual improvement team should capture, document, assess, and prioritise the ideas and act appropriately. [4]

Project management is an essential part of the thesis. There is **project management** among the general management practices. Since there is a dedicated section (see Section 2.2.1), we will skip its description here. On the other hand, we will continue with the management practices, which we might use to direct the project and improve our approach in general.

Software development and management will be the first technical management practice which we will go through. Its goal is to ensure that the applications meet the stakeholders' needs. Specifically, this practice ensures application functionality, reliability, maintainability, compliance, and audibility. The software development and management include all the design and development activities, such as design, development, testing, safekeeping preparation, deployment, and version control. Naturally, not all projects will include all the activities. [4]

Service design is a service management practice. Its purpose is to design the products and services to fit its objectives while deliverable to the organisation. The service design is closely related to project management and includes planning and organising new and changed products and services. It also includes planning and organising the interaction between the organisation and its customers, but that is not that important for the help desk. Services and products can be missing some functionalities, too challenging to use, or too expensive to run. The service design also focuses on customer and user experience to make the service as pleasant as possible. It is therefore essential to:

- perform the design activities with the customer,
- have the design methods understandable and quick,
- ensure that the product or service will be maintainable and cost-effective, and
- consider the whole environment of the product or service. That includes:
    - all organisations in the chain,
    - other products and services,
    - architecture,
    - technology,
    - management practices, and
    - measurements and metrics.

It is possible to use service design during the whole lifecycle of the asset, even when retiring a product or a service. On the other hand, not all products and services demand the same degree of attention. There is an approach called design thinking presented in [4]. The last three steps of the approach are prototyping, implementation and evaluation. We are interested in those steps because of their use during the development. We can see similar steps in the improvement management, but the prototyping is new. [4]

Information security management is a general management practice. Its goal is to prevent, detect, and correct security incidents. On the other hand, it must not lock the information from those who need it in the name of information security. The information security management also needs to think about improvements and allow the organisation to develop. The same problem has the **risk management**, which is another general management practice. Its purpose is to identify, assess, and treat any risks if possible. We will look closely at the risk management in Section 2.2.2. There are no specific methods how to secure the information in [4]. On the other hand, it emphasises the importance of audits and reviews, risk management processes, incident management processes, event management, and procedures for testing and managing changes. [4]

Release management is another service management, one of those practices that might help us with the project. A release is a version of a system that is available to use, and the purpose of the release management is to make those releases available. It is closely related to project management, so we will use some terms we have not defined yet. Release management has different structures depending on the project management method. In the waterfall management method, the deploy and release go in hand, and all changes and documentation are released simultaneously. In the agile management type, each task ends with a deployment. After the last deployment, the next step is to release the change. The time for a release must be agreed on with the stakeholders beforehand. A release schedule documents the timing of the releases. Generally, the first release is for a small group to ensure it works correctly. If it does, other groups can access the release. This approach applies to both types of project management. The organizations often use the blue/green release or feature flags of the staging. The feature flags is a method, which enables to release of specific functionalities to specific users. The blue/green release is a method in which the users can switch between two system versions. One in testing and one successfully deployed. [4]

Service validation and testing is a service management practice. We touched testing of systems already in the release management. The goal of this management practice is to ensure that the deployed products and services meet the requirements. The validation part of this practice management sets the criteria for release management, which means that service management sets under which conditions a system can be released, and the release management plans it out. The final piece is the testing. It is this management practice, which tests the system and says whether it met the release criteria. There are four typical types of functional tests. The unit test tests a single component. The Integration test tests a set of the components connected by something, for example, a function or a database. Finally, the system test tests the whole system. The last type of test is the regression test which tests whether previously working functions are changed in any way. There are also different types of non-functional tests, which are essential too (see, e.g. [4]). [4]

After the service is tested and released, it is time for **deployment management**. It is a technical management practice whose goal is to move the system between development, testing, and live environments. There are four typical ways to deploy a system. We touched the phased deployment in the release management. In this type of deployment, only a group of users get the new or changed component. That repeats until all users have it. Continuous delivery is the next level of phased deployment. The components undergo the whole process when they need to, so there are many opportunities for feedback. That is ideal for agile project management. The “big bang” deployment, on the contrary, is typical for waterfall project management or databases changes. In this approach, all users get the new or changed component at the same time. The pull deployment provides access to the new or changed system for all users but does not require them to download it immediately. The communication around the deployment is part of the release management. The deployment management should focus on the components themselves. That would include safekeeping it at multiple locations if anything happened to one of those.

From now on, we will look at those management practices, which we will use during the design. That means that they carry some functionality, which we will implement. There were already some functionalities in continual improvement, information security management, and release management. There are also some in risk management (see Section 2.2.2).

The **service desk** is a service management practice. According to [4], its purpose is to gather incident resolution and service requests. It should also be the communication channel for the service providers and their users. Today, service desks can help with more than technical incidents. It is possible to use automation for problem resolution and chatbots to increase customer experience and ease the work of the operators. The service desk should provide different communication channels like:

- **calls**; phone calls, conference calls, and others,
- **chat**; live chat, and chatbots on service desk channel, social media, or forums,
- **email**; mostly automatic for logging updates and confirmations,
- **personal**; walk-in service desk is sometimes necessary and even more helpful, and
- **access to databases**; through service portals and mobile applications.

It is essential to train the service desk staff, because of two reasons. First, they need to provide specialised services. Second, they will directly influence the customer experience. In [4], there is a list of specific services of a service desk. Those are:

- **phone service**; intelligent telephony systems, incorporating computer-telephony integration, IVR, automatic call distribution, call recording and quality control
- **workflow systems** for routing and escalation,
- **workforce management** system,
- **the resource planning** system,
- **knowledge base**,
- **remote access** tools,
- **dashboard and monitoring** tools,
- **configuration management** systems.

Service request management is the service management practice in which the service consumers ask the service provider to provide the service. The help desk case includes a request to deliver, provide, or access a service, resource, or information. It also includes feedback. A request might include multiple of those requests. Requests are not incidents; they are a normal part of service delivery. On the other hand, they might become an incident if the organization ignores them. Since requests are a normal part of service provision, the steps taken to solve it should be known before. They might be prepared and automatised to ease their resolution. Some might be fully automatised, like information requests. Others cannot be automatised at all because of their complexity. On the other hand, the organization should automatize them to the highest degree possible. It should be clear which processes require little or no approval. There should be clear outputs of the services, so the users know what to request and expect. [4]

Incident management is one of the reasons for building a help desk. It is a service management practice whose goal is to minimise the negative impact of any incidents. An incident, in this case, means an unplanned stop of a service or a reduction of its quality. It is necessary to restore the service to the normal as quickly as possible. Priority management is vital for good incident management, and the solutions should be collected using knowledge management. The improvement management should also collect data about the incidents because there might be things to improve. The people working on the incident should provide regular updates. Those should contain symptoms, business impact, configuration items (see Service configuration management) affected, actions completed, and actions planned. Each of these should have a timestamp and information about the people involved. Incidents will require a different approach. The users can resolve some on the spot, but some will need a support team to resolve. For effective incident management, teams need to cooperate, and there should be a process for knowledge sharing. All the incidents should be logged, managed, and investigated. That is for a more straightforward resolution in the future. [4]

IT asset management is another initial reason for this project. It is a service management practice whose purpose is to plan and manage the lifecycle of IT assets. IT asset is anything of a financial value that helps deliver the IT product or IT service. It helps the organisation control costs, manages risks, support decision-making about the assets, meet requirements, and maximise value. Asset management, in general, is not only limited to managing the lifecycles but also to the acquisition, operation, care, and disposal of the assets. Software asset management, on the other hand, is a sub-practice of IT asset management. It aims at management, control, and protection of software in the organisation throughout its lifecycle. The asset register stores the inventory information. This register is generally first filled by an audit, but then, it is better to gather the data whenever there is a movement of equipment. That means a purchase, disposal, repair, or whenever any data about the equipment is changed. Then it is possible to conduct the audits with a lower frequency. Since asset management stores data about both hardware and software, there are different requirements for them. The hardware needs an identification label. Hardware might need special care when re-using or thrown out, mainly because of security or regulations. It is necessary to protect both the software and the hardware from theft. In the case of software, it is crucial to keep a verified proof of purchase. It is necessary to track which devices have access to cloud-based assets and regularly update the permissions. There should be an individual who will take care of assets given to clients and perform processes if a device was lost or stolen. Combining the system or connecting it with a system for service management and financial management is good. If they are not combined, it is good to use the standard naming convention. The IT asset management connects to the **infrastructure and platform management**. That is a technology management practice and focuses more on overseeing the assets rather than categorising them. [4]

Knowledge management is a general management practice. Its goal is to maintain and improve the use of information and knowledge across the organisation. The main areas to improve are effectiveness, efficiency, and convenience. Knowledge management should provide a way to collect, store, and re-use knowledge. Knowledge, in this case, means information, skills, practices, solutions, or problems. On the other hand, knowledge must be usable and relevant. That means that a 1000-page-long manual is useless for a technician who urgently needs to repair a printer. A set of instructions or reference points are much better. Knowledge management should provide the correct information and the right time, in a proper format, at the right level to the stakeholders according to their access. [4]

Problem management is a service management practice. A problem is a cause, or a potential cause, of an incident, and problem management is the one practice handling this. That includes identification of the problems, their control, end in some cases, error control. Error, in this case, is a problem, which has been analysed but not resolved. A help desk can help the problem management in all its phases. A problem is a cause for one or more incidents. The incidents are taken care of by the incident management. On the other hand, the problem management tries to solve the source of those incidents. That might include workarounds or long-term resolutions. The problem management must analyse the incidents to identify a problem. They can use, for example, trend analysis or detection of duplicates. On the other hand, it might be more beneficial to analyse the high priority problems in-depth rather than analyse all the minor problems. The problem management must document the problems and their solutions though they resolve them once and for all. It is helpful to adopt the risk management techniques for problem management. When the problem management finds a problem solution, improvement management and change enablement should take over. The CIR should contain the problem and the solution, so the problem management should cooperate with the knowledge management. [4]

Availability management is a service management practice. Its purpose is to ensure that a service is available when it is needed. It is best explainable on an example of service inside an organisation. Let us assume that the employees in a division in the company provide service to other divisions. Other employees need their help during the workday. An availability management system ensures there is always someone in the division, even though some personnel went on vacation. The contract, which the parties agreed on, contains the timeframe for the employees in the workplace. The same applies to any service, like, for example, email, MS Office, or a gym. The availability management must also ensure an infrastructure to plan out the availability and non-availability of a service. They should also monitor, analyse, and report on the availability. The analysis commonly looks for mean time between failures (MTBF) and mean time to restore service (MTRS). On the other hand, the reports also might include other measurements like user outage minutes. The availability influences user satisfaction. **Workforce and talent management** meets the goal of availability management in some parts. Part of this general management practice's purpose is that the organisation has the right people at the right place. In this case, the workforce and talent management cares more about knowledge rather than shift management. On the other hand, they meet when there is a problem requiring special skills. Besides that, talent management has some features, like recruitment, or personal development. [4]

The workforce and talent management might be an inspiration for the future improvements of the help desk. There was one more management practice, problem management, which might inspire too. In the following paragraphs, we will look at other management practices, which might inspire new functionalities.

Monitoring and event management is another service management practice. It is a practice thanks to which the organisation observes, record, and report on their services and service components and changes. It assesses the service value system and touches the incident management by responding to the events that indicate incidents. Automation is said to be the key to successful monitoring and event management. It is essential because there might be many data, and it demands correlation between the events. [4]

Service catalogue management is a service management practice. Its purpose is to provide a single source about the services and their offerings to the relevant audience. That includes the services inside the organisation. The information is for users, who need to know what they can request, and others (see, e.g. [4]) who will not be service consumers for the help desk. The **service configuration management**, which shares the information about the service's configuration items (CIs), is closely related to that. A CI is anything needed to deliver a service, for example, an application, equipment, or a room. This service management practice has other functions, but they are not essential here (see, e.g. [4]). [4]

Change enablement is a service management practice. Its goal is to manage change projects to their successful ends. There are three types of changes: priority, riskiness, planning, and needed authorisation. On the help desk, there might be a module for authorisation of processes, which might help this practice, and in further updates, the development team might implement the whole process. See [4] to read more about the practice. [4]

The evaluation team might use the **measurement and reporting** general management practice for an evaluation. There were already some practices, like availability management or service validation and testing, which they might use. The purpose of the whole practice is for the management to collect data to support sound decision making. It describes functionality which we will not go through but might be interesting for further development of the help desk. On the other hand, it also describes how to track and evaluate a project. Success is generally measured, for example, by profit, growth, customer retention, or operational/public service. The organisation must convert the goals to critical success factors (CSFs) to evaluate a project, which gains its value using key performance indicators (KPIs). A KPI is a metric, while CSF is a precondition for achievement. The team might use a list of KPIs in [41] as inspiration when developing one. It is important not to focus only on one KPI since it might lower the performance. On the other hand, KPIs might work like guidelines, so if there are too many KPIs, the employees might get confused about what to focus on. [4]

2.2. A project

Now, since we can describe an organisation and its services, we can improve further. The improvement is a project. It has a goal, a process, a team, and management. The goal is to improve something in the company. The process is a set of tasks that end up with a satisfying goal. Those tasks are specific for each project and will be defined later in the thesis. There are some constraints for the tasks. That is time, cost, and scope. [30] [37]

The team is an organisation or a set of organisations that completes the project. It is possible to have different sizes of the team. The roles are also not predetermined, and some teams might be missing a few roles. The team structure directly relates to project management.

2.2.1. Project management

Project management is a process carried out by the project manager/s. It is continual, and it helps the team plan, organise, and manage the tasks within the constraints. They must define the constraints at the start of the project, but they can change during it. The main objective of the project management is to direct the team to accomplish the set goal/s. Project management should not be reactive but somewhat proactive. [30]

The project management has a similar structure as the improvement management. We must first understand what is wanted and what are the constraints. Generally, a customer provides a project specification, which contains most of the constraints. On the other hand, some projects need comprehensive specifications. They might need to be analysed, and the team might want to raise some questions for further specification for good project management. Projects, especially the big ones, need a work breakdown structure (WBS). We will look at WBS closer in the next section. [30]

The projects in controlled environments (PRINCE2) (see, e.g. [57], [38]) are one of the most widely used methods for project management. It joins the best practices from many different projects around the world. Thanks to that, PRINCE2 is project type independent. Most of what we will look at in this chapter relate to PRINCE2.

There are four basic approaches to project management (see Table 2) – waterfall, iterative, incremental, and agile. The waterfall project management approach is also called predictive. That is because of its dependency on stable requirements and low risks. It executes the tasks in sequence and does not return to them, which creates a waterfall in its time plan. The team usually deliver the product at the end of the project. There is no value for the organisation during the project. [32]

Table 2: Characteristics of different project management approaches (source: [33])

Approach	Requirements	Activities	Delivery	Goal
Waterfall	Fixed	Performed once for the entire project	Single delivery	Manage cost
Iterative	Dynamic	Repeated until correct	Single delivery	Correctness of solution
Incremental	Dynamic	Performed once for a given increment	Frequent smaller deliveries	Speed
Agile	Dynamic	Repeated until correct	Frequent small deliveries	Customer value via frequent deliveries and feedback

The iterative project management approach builds on the waterfall approach by repeating the activities until they are acceptable for the customer. Therefore, the requirements change in time. On the other hand, the iterative approach still does not go back. That means that the time plan still looks like a waterfall, except the length of the phases can vary in time. [32]

An iterative life cycle prioritizes the speed of the project. Not all projects can have an iterative life cycle since it delivers partial solutions to the customers. On the other hand, it still delivers the value only after the final iteration. [32]

The agile project life cycle combines both iterative and incremental. It is also the development methodology in ANS CR. Agile project management divides the goal into multiple deliverable systems. Each iteration delivers a new part of the system. Furthermore, the team repeats the iterations until the customer is satisfied. We will go through agile in more depth in Section 2.2.3. [32], [37]

According to [4], it is best to use the waterfall method for a project where the requirements are known, and it is improbable that they would change. The waterfall method is best for defining the work rather than speed or delivery. The agile method is suitable for uncertain projects and where the product and customer are more important than the definition and sticking to the plan. The iterative and incremental methods are rare, and they are instead evolution steps from waterfall to agile. [4]

We have already mentioned the time plan. That is one of the tools, which managers might use. A popular scheduling tool is the Gantt chart. That is a list of tasks extended by the time plan. The time plan has a horizontal axis, broken down into a time unit – typically days, weeks, or months. A line for each task in the time plan represents when it starts and how long it will last. It is even possible to break the tasks down like the WBS and add a time plan for those sub-tasks. For clarity and better visibility, the tasks generally have different colours. We will look at the Gantt chart for agile development in Section 2.2.3. To read more about it see e.g. [39], [40]. [30]

Another task for project management is risk management. The team management must first identify the risks, then analyse them, and in some cases, assess the risks. This process is critical for project management to be proactive. Risk management aims at lowering the impact of possible risks at costs, schedule, or design. Each risk should have its code, name, probability, and impact. Especially for longer projects, it is essential to monitor the risks and report on them continuously. There are several tools and techniques for risk management. We will look at risk management closer in Section 2.2.2. [30]

2.2.1.1. Work breakdown structure

WBS is a division of the work into groups that are easier to complete. For example, writing one section on one topic ten times is more manageable than writing a whole book. The tasks might be broken down even further, making minor tasks. [31], [30]

Four concepts should be kept in mind when creating a WBS. The tasks should be hierarchical, meaning they should have defined levels of complexity and depth. The tasks should be a decomposition of the task or project above. That means that a subtask should never be more complex than a task. All the tasks should follow the scope of the project. Moreover, all the tasks should be deliverable. Meaning that there should not be a task, which we cannot accomplish in the constraints of the project. [32]

There are also other rules for the WBS mentioned in [32], such as:

- The tasks should not contain cost.
- The tasks should not imply importance.
- There is no limit to the decomposition of the project.
- The decomposition should lead to a division of the work. Meaning there should be more than one subtask under a task.
- The tasks do not overlap, create relationships, and there are no duplicates.
- Tasks do no assign resources, time, nor sequence.
- Numbers and dots between them should number them. The amount and value numbers depend on the number of the upper task and the level of the task.
- The project is in the first level. If there are multiple projects, there can be more than one number.
- Level two are the high-level breakdown of the significant areas of the project like:
    - deliverables,
    - project phases, or
    - management.
- Other levels are generally more decomposed tasks. The work packages are at the lowest level.

The waterfall project management is typically just as described by the above bullets. Generally, it has three levels, but it might have more depending on the complexity. The difference between waterfall and agile in the WBS is the first two levels. The first is called a product rather than a project. The second might be different in other ways than just terminology. The second level of agile project management tasks is either iteration, releases, features, or epics. The iterations are always one level above the user stories and represent what activities will the team repeat to achieve the set milestone. Features are the milestones or the functionalities of the system. The release is either feature or a set of features. The release should function independently, so if some functionalities depend on each other, they must be in a release together. The epics are minor projects which together create the product. [32]

There are different methods for the WBS creation in [32], and each carries some advantages and disadvantages. Each also has its steps or workflow. Table 3 summarises all the information about those methods. It is common to have a brainstorm while creating the WBS. To lower the risk of losing some ideas, we can use mind maps while completing the tasks and subtasks. Twenty questions in [32] might help the team decompose the project.

Table 3: WBS creation methods (source: [32])

	Advantages	Challenges	Approach
Top-Down	convenient for status reporting	Requires constant attention for overlooked work packages.	Identify the final product(s), service(s), or result(s) of the project.
	logical		Define the project's major or interim deliverables.
	valuable when brainstorming deliverables		
	flexible deliverables	Elaborate WBS sufficiently to permit management oversight and control	Decompose major deliverables until management agrees.
	Creates the product roadmap		Review and refine until stakeholders agree.
Bottom-Up	It starts with all deliverables or user stories and works backwards into a project.	Identify all deliverables or work packages before producing the WBS	Identify all the deliverables (work packages or user stories) involved in the project.
			Logically group deliverables together.
		Group work packages logically	Aggregate deliverables further.
	Ensures inclusion of all work packages or user stories.		Analyse until all the work has been encompassed.
		Lose focus on the big picture	Repeat until everything aggregates to the project.
			Review and refine until stakeholders agree.
Standards or Templates	Leverage predefined formats.	Make a project fit the standard.	
	Enhances cross-project WBS consistency.	Not all projects fit into a highly structured set of WBS standards or templates.	
	Facilitates the implementation of principles and good practices.		
	Provides a starting point for WBS creation.	Requires a project to fit the standard.	
	It helps determine the appropriate level of detail required.	It might include unnecessary WBS or miss some WBS.	

Finally, we can create a diagram or a table for all the projects, tasks, and subtasks. There are three ways how to represent the decomposition of the project in [32]. The hierarchical representation is the best diagram for presentations to other people or the physical world (see Figure 4). This representation is used, for example, in [34]. The outline style is somewhat confusing for the human eye in its basic form.

On the other hand, it is helpful for computers in its tabular form. Such a table contains columns for level, code, and name. On the contrary, we would use the tabular outline to have a column for each level.

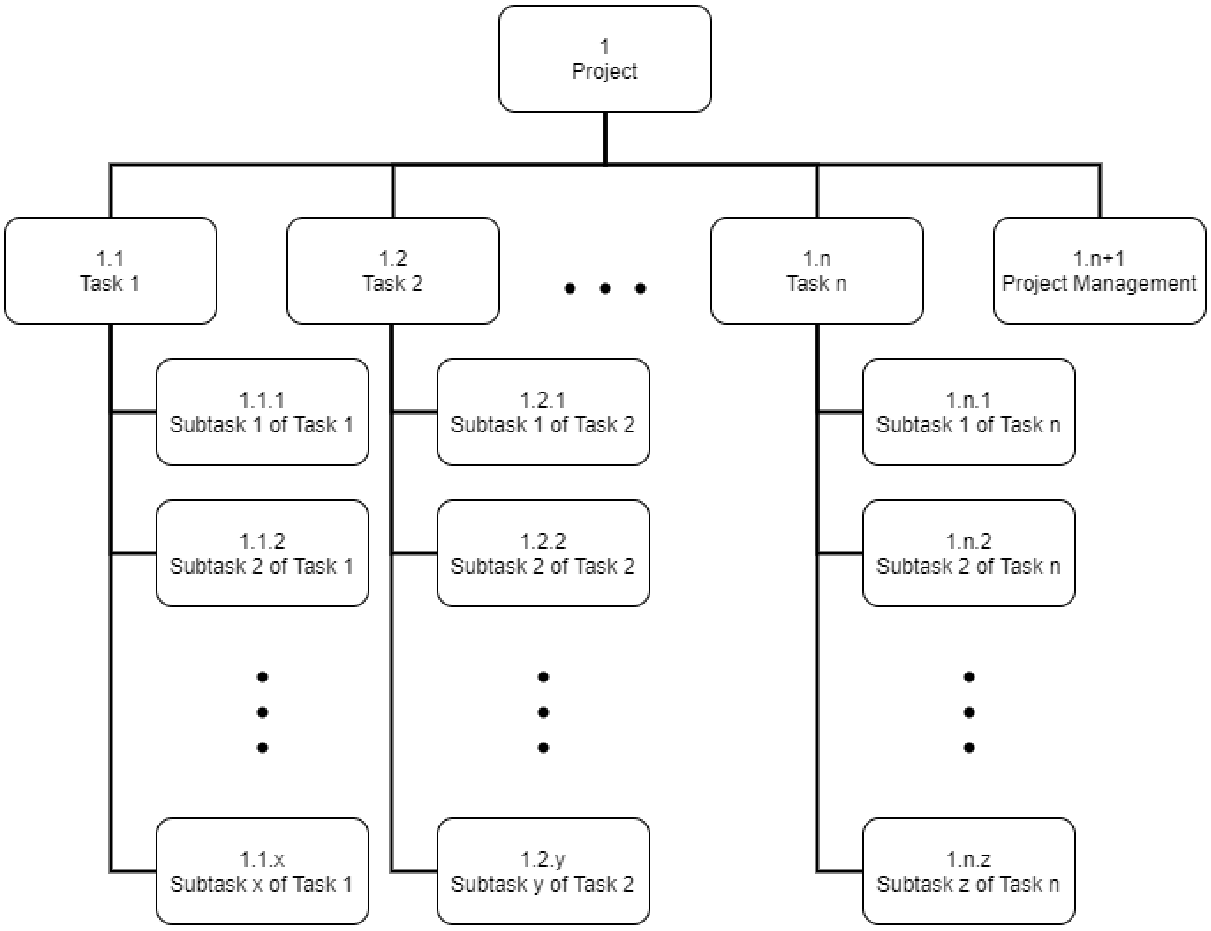

Figure 4: A WPS hierarchical structure

2.2.2. Risk management

Risks are generally something that could endanger achieving the goal. They occur both in projects and organisations. A risk consists of the likelihood and impact of adverse events. That is the techno-scientific perspective on risk, and we will use this definition. There are different definitions of risk from different perspectives in [50]. The organization generally mitigates all the risks unless it is too costly or too restrictive for further development. [4], [47]

Risk management is an integral part of both a project and a whole organisation. It is a general management practice for organisations. Organizations generally avoid the risks. On the other hand, they should not and cannot avoid all risks. Some are simply too complex or unpredictable that it is impossible to prevent them. Some relate to opportunities, and the organisation must embrace few to be competitive and innovative. Naturally, not all opportunities are worth the risks. That is why the organisation must identify, assess, and, if possible and desirable, threaten the risks. Organisational risk management is similar to project risk management, which we will go through in the following paragraph. On the other hand, it is also more general and focused on all services. See, e.g. [51], [52] to read more about organisational risk management. [4]

In the waterfall project management methodology, it is easy to perform risk management. That does not mean there are no risks in the agile project management methodology. Since project management is the first step for waterfall projects, it is possible to assess the risks initially. The agile project management methodology's requirements are flexible, and the development is quick, making it hard to mitigate the risks. In [45], there is a literature review on risks in agile project management methodology for teams spread across the globe. According to [47], organizations often neglect risk management in the agile project management methodology. The authors surveyed the most common security risks and how to mitigate them.

This project is about design and partially about the development of software. The [44] and [48] discusses the risk management for the software development projects. The authors divide risk management into multiple steps. In the first half of the process, the risk management teams should identify assets, threats, and vulnerabilities. In the second part, they should determine the likelihoods of the events and their impact. Then calculate the risks, mitigate them, and, finally, create plans and strategies for risk management. The author of [53] presents the seven most common risks in software development projects. While searching for risks, we can also use [54] for the types of risks.

After a project, the team should extract all the risks and save them in the knowledge base. There is an example of a risk statement in [48]. The knowledge base can then help in other projects with identification, assessment, and mitigation. The team can create the risk statement before the end of the project and fill it during it, which eases the workload during the project. Finally, there is a self-assessment checklist, which helps risk management.

Risk assessment is a big part of risk management. It is the step in which we calculate the risks from the likelihoods and impacts. There are different tools for risk assessment (see, e.g. [48], [55]). One of the methods for risk assessment is called fault tree analysis (FTA). It is a simple method in which we create a tree of events with probabilities and outputs. There are different ways to construct an FTA (see, e.g. [55]), but the product is an explicit situation model.

There are different ways how to mitigate risks. It highly depends on the risk. On the other hand, there is some general advice on the subject (see, e.g. [48]).

2.2.3. Agile development

Agile development is a set of methodologies that focus on the product, flexibility, simplicity, and collaboration. While using agile development, we must also use agile project management. During the development, the team meets the customer frequently and delivers working products during the whole project. The customer might change the requirements during the project depending on the delivered product. The team management then must change the planning accordingly. The team structure is also, in comparison to the traditional groups, flat, which means that all the team members are equal and the management lead by example rather than authority. Since its interception, it has been widely adopted in IT development, business, finance, medicine, or education. [35], [36]

The waterfall methodology has a linear and serial Gantt chart, which makes it clear. There is enough time for each step to complete it correctly. At first sight, we can see that it will deliver something functional only at the end of the project and not a second earlier (see Figure 5). On the other hand, the customer must wait long for the product. The stakeholders can enter the project only at the beginning and then get the result. On the other hand, the agile Gantt chart might seem like a mess (see Figure 5). The team carries out all the tasks multiple times. That is because the agile methodology delivers the final product piece by piece. [36]

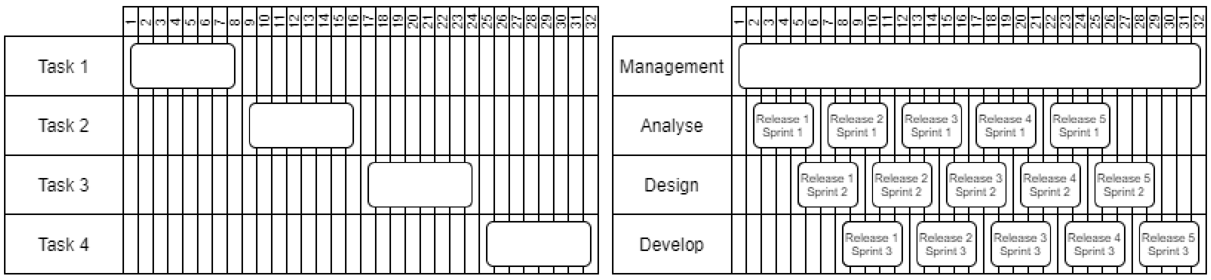

Figure 5: Gantt charts (waterfall on left, agile on the right)

Since [36] is written by the same authors as [4], agile can use principles from ITIL. We can see that the projects' broader context is the best, which means that each project is part of an improvement or a service value system. Vision consists of the products or projects which the organisation will do or is doing. Those products consist of releases, which compose of sprints. [36]

2.2.4. Requirement gathering

Requirement gathering is an integral part of project management (see, e.g. [9]). We can collect interview requirements using focus groups, facilitated workshops, questionnaires and surveys, observations, or prototypes. There are different techniques for those approaches, but their efficiency depends on the project and the manager. In [56], there are steps for successful requirement gathering. [9]

The output of this process should be requirement documentation, management plan and traceability matrix. The documentation should contain functional, non-functional, and quality requirements and constraints. Each requirement should have its acceptance criteria, impact, and assumptions. We create a requirement management plan to decide how to handle changes in requirements beforehand. Finally, the requirement traceability matrix is for track the requirements. It shows the fulfilment of the requirements and which requirements can the team exclude at the expense of others. [9]

Since the interviews are essential for the business analysis, connecting the requirement identification with the modelling (see, e.g. [55]). We will look at modelling in Section 2.3.

It is possible to use the use cases for capturing the functional requirements. They describe what the system should be able to do. The non-functional are generally not mandatory but desirable. They describe how the product should work and define the product properties. The quality requirements are again not mandatory. If fulfilled, they raise the user and customer experience. The constrains self-explanatory. They, for example, define other systems with which the product must communicate. [9], [64]

2.3. Analysis and modelling tools

A system is a set of parts that together exhibit behaviour or meaning, which the individual constituents do not (see, e.g. [58]). The designed application in this thesis is a system too. For most systems, it is possible to create a model. Such models generally have input, processes, output, and sometimes feedback (see, e.g. [59]). Model-driven design and development became very popular. The most common language for model-oriented development is unified modelling language (UML). According to [61], the model-based design could benefit the development. Before the design and development, there should be an analysis conducted. [4], [60]

2.3.1. SWOT analysis

SWOT analysis is a tool to evaluate ideas or projects. SWOT stands for strengths, weaknesses, opportunities, and threats. We generally use the analysis to organise information and identify issues. On the other hand, it should also produce solutions or opportunities. The analysis should show the connection between the internal factors, strengths and weaknesses, and external factors, opportunities and threats. It is good to perform the analysis for both existing and new businesses. The existing might learn about possible improvements, while new businesses should use it as planning and development. The draft for the SWOT analysis is generally a 2x2 table, and each square is then one of the words from SWOT. Meaning that the top right square is for strengths, the left top square is for weaknesses. The bottom squares are for the opportunities and threats. It is better to perform the analysis in a group. [42]

2.3.2. Unified Modelling Language

UML is a set of diagrams and elements with defined meaning used for creating the models. We can use the different diagrams for different views in the model. The definition of the meaning of all the elements is vital for us because of its universality. If we create a model using a spoken word, someone might misunderstand us, while the UML ensures that everyone understands the model. If we use model-driven development, we can convert the model into a programming language afterwards. Thanks to that, UML is suitable for any part of the development, like specification, visualization, architecture design, construction, simulation and testing, or documentation. [7], [8]

There are two types of diagrams in UML. We use structural diagrams to show how objects, components, classes, and packages relate to each other. The behavioural diagrams, on the other hand, display the functioning of the system. Though there are different elements for each diagram, UML allows their usage elsewhere. Because the elements might appear in different diagrams, there are numerous different diagrams. There are 13 of the basic ones, but we will look at only some of those. [8]

The **class diagram** is a structural diagram. There are two main elements within the class diagram: the classes and the relationships. The classes represent a real-world family of objects. For objects to create a family, they must have similar structure, behaviour, and meaning. It encapsulates the attributes and methods of those things. The relationships are then of multiple types. We use generalization for the extension of classes. The child class has all the attributes of the parent class and probably some new ones. We use the association if the classes interact with each other somehow. There are more relationships and elements, but we will skip them because we will not use them. The class diagrams are derived, and therefore similar to, the entity-relationship diagrams (ERD). We will talk about ERD in Section 2.3.3, but the main difference is that class diagrams have methods, and ERD are only for data models (see, e.g. [75], [76]). Another difference, which we have not talked about yet, is access. In the class diagram, we can indicate which attributes and methods are public, private, or protected in the class. We use symbols before the names of the attributes and methods to indicate that. Plus means public, minus private, and hash means protected. [5], [6], [7], [8]

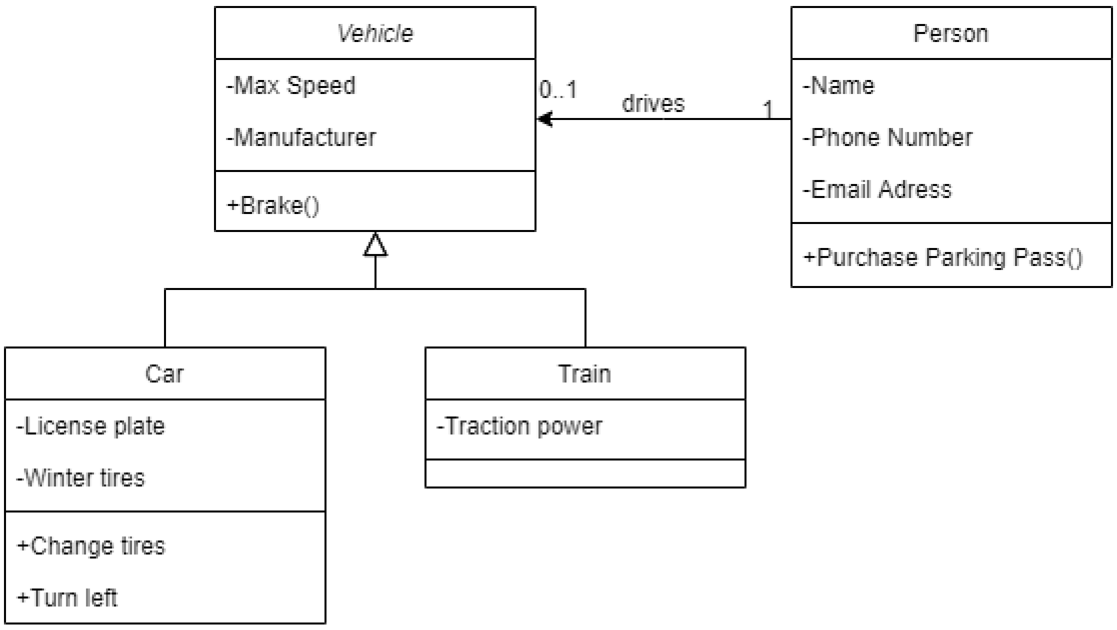

Figure 6: Example of a class diagram

The **use case diagram** is a behavioural diagram. We can use the use case diagrams to show the functionalities of a system and the participants on the functionality. It does not describe the functionalities any further. The main elements of use case diagrams are:

- system,
- actors,
- use cases, and
- relationships;
    - association,
    - inclusion,
    - extension, and
    - generalization.

The system is the boundary in which there are functionalities. The actors are the people or systems which play a role in the functionalities. Though the system itself generally participates in the processes, too, it is not an actor. The use cases are the functionalities. The association is the relationship between the actors and use cases. They indicate the participation of the actor in the functionality. We draw, include and extend relationships between use cases. The include relationship indicates that the parent use case runs whenever the child uses case runes too. The meaning is not mutual, so the parent can run even though the child is not running. We use the extended relationship if the children use case might run whenever the parent runs. The generalization is for both the use cases and actors. We use it only if the child has additional properties in the diagram. Use case diagram is excellent for its analysis value. For coders, other diagrams are much more important and exciting. [5], [7], [8]

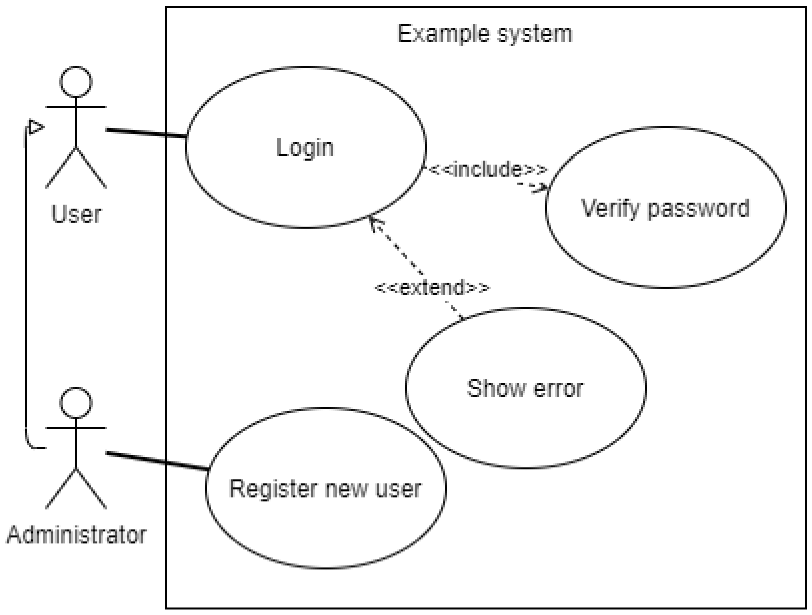

Figure 7:Example of a use case diagram

We use the **activity diagram** to show processes across classes. We can see all the operations in the class diagram unless it is an outside actor, but we cannot see the sequencing. We will look at some of the elements it composes of, such as:

- activities,
- flow, and
- control nodes;
    - initial and final,
    - decision and merge, and
    - fork and join.

The activities are composed of actions, which are the most straightforward piece of behaviour. They can be in the diagram too, but we will not be using them. The activity is a step in the process. The flow shows how the activities and control nodes are sequenced. Whenever we divide it into multiple flows, we should join them together afterwards. The initial and final nodes are self-explanatory. The final node can sometimes be also an activity, but we will not be using that. The initial activity should generally be in the top left corner, while the final should be in the bottom left corner. On the other hand, it is again not necessary. Also, there should generally be only one initial node and only one final node. However, some additional elements may contradict this statement. A decision is a condition element that has two output flows, one being true and one false. Sometimes, the decision has more outputs describing one situation, but we will not be using this. The merge then joins the decision flows back together. We should have the same number of merges as decisions. Some designers exclude this element, but we will use it. Finally, we use the fork to indicate activities going in parallel. There can be as many flows going from the fork as we want to, but they must join back together by joining. There is another element that is common in activity diagrams. It is the swim lane. This element is in the diagram's background and shows which actions and control nodes belong to which actors. [5], [7], [8]

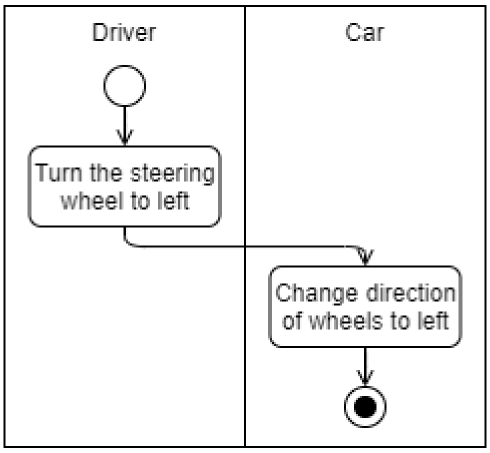
Driver
Car
Turn the steering wheel to left
Change direction of wheels to left

Figure 8: Example of an activity diagram

The component diagram is used the most by the coders. Business analysts connect the requirement gathering with the use case diagrams (see, e.g. [55]). The coders, on the other hand, convert requirements directly to component diagrams. After they complete the component diagrams, they generally move to the deployment diagrams. We will skip their description here because we will not use them in the thesis. [7]

There are many suitable applications for creating UML. For simple projects, we can use draw engines like diagrams.net (see [71]) or Lucidchart (see [72]). For more complex projects, it is crucial to use model-driven architecture when creating diagrams. For example, there is StarUML (see [73]) or Enterprise Architect (see [74]), which are great applications for creating models with UML.

2.3.3. Entity Relationship Diagram

The entity-relationship diagram (ERD) is a data modelling tool. We use it to describe and portray data. We can use ERD to model new data structures or analyse current ones. The first step in creating an ERD is to create a conceptual model and then optimize it into a schema called the ERD. [77]

There are two main ways how to draw an ERD. We will use the one called “crowfoot”. In this type of ERD, there are two main elements:

- entities, and
- relationships.

The entities are tables that contain the data. They represent a system about which we wish to store data. On the other hand, we will shortly learn that this definition might not apply to all entities. Along with the name, the entities have attributes and instances. We use the attributes for different data about the entity. We cannot we the instances in the ERD, but they are the rows in the table, the data, while the attributes are the columns. [77]

Each entity must have its primary key. That is a unique identifier for each instance. Generally, we accomplish this using identification numbers (ID). They are integers, and when a new instance occurs, its value raises by one and bounds to the new instance. On the other hand, it is also possible to have a different primary key if unique. Though it is better to use one attribute as the primary key, it is possible to use multiple attributes. [77]

Finally, the relationships connect the entities. They do so through the attributes. There are different types of relationships. They all represent the same, but the multiplicity might change. There are:

- 1 to 1,
- One to many, or
- many to many relationships. [77]

We do not use the 1 to 1 relationship often because it indicates that we can join the entities together. The “one to many” relationship is the most common, and basically, there should be only these relationships in the ERD. Many to many relationships are common, though there cannot be any in the ERD. We must remodel the “many to many” relationship into two “one to many” relationships and new entity. There are also variants of those relationships stating whether the attribute is nullable. That means whether there can be an instance with no value in the attribute. [77]

The relationships take a primary key from one entity and use it in the second one. That creates a link from one table to another. That creates a foreign key. Depending on the situation, if there are multiple relationships to an entity, there might be multiple foreign keys. [77]

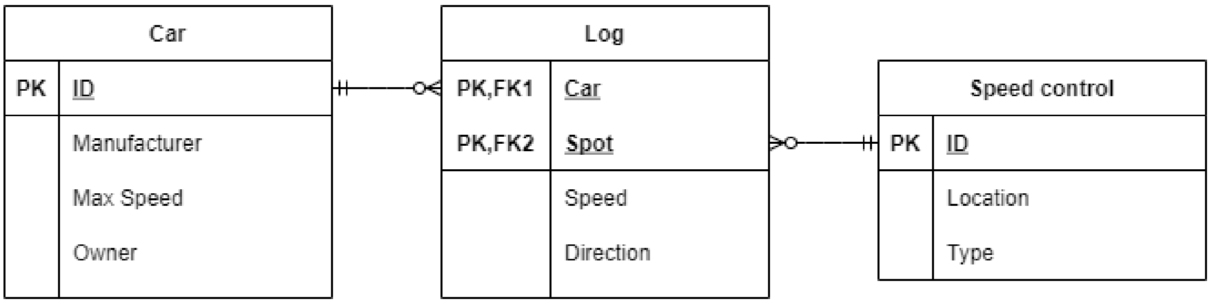

Figure 9: Example of an Entity-Relationship Diagram

3. Air navigation services of the Czech Republic

Air traffic is a complex system. It differs from ground traffic in many ways. The most notable one is one more dimension of free movement. That makes air traffic more flexible. Unlike cars, planes need more than one pilot. There is not only a crew for the operation of the plane but also air traffic controllers (ATCOs) and flight operations officers (FOOs) (see, e.g. [14], [15]).

Air traffic control (ATC) is a service provided to transportation companies. It ensures the safety of air travel. Thanks to radars, ATCOs have an overview of the airspace and using radio, they can communicate with the crews of the planes. They ensure no collision in the air by optimising the planes' speed, direction, or flight level. [16]

Countries manage ATC. That means each state has its own ATC centre or cooperates with surrounding countries (see, e.g. [17]). Civil Aviation Authority of the state must licence organisations providing ATC. In Czech, the license belongs to the Air navigation services of the Czech Republic (ANS CR).

ANS CR exists since 1995, under the resolution of the Ministry of Transport and Communications. Commercial Register at the Municipal Court in Prague; section A, entry No. 10771 records ANS CR. [18]

In the year 2019, ANS CR had a turnover of almost 4 billion Czech crowns. There were more than 900 thousand movements in Prague FIR only. The Coronavirus hit the air traffic the most. Though ANS CR has not published the annual report for the year 2021 yet, it is safe to assume that everything has fallen.

On the other hand, EUROCONTROL periodically publishes statistics and insights inside the air traffic. The latest statistics show that air traffic is on the rise again. According to [19], the current number of flights is the highest since the crisis began.

According to [18], the vision of ANS CR is: "*ANS CR is a dynamic, stable and reliable, socially responsible and sustainably developing component of civil aviation in the Czech Republic, actively contributing to its further development. At the same time, it is a selfconfident element of the integration and liberalisation processes in European ATM environment, where its overall value and competitive ability will further increase. In order to participate in the liberalisation processes effectively and actively, the company strives to achieve the standards best suited to international integration efforts.*"

3.1. Air navigation services of the Czech Republic as an organisation

We can describe the Air Navigation Services of the Czech Republic (ANS CR) as an organisation using an Information technology infrastructure library (ITIL). We described how we could model organisations using ITIL in Section 2.1.2.

ANS CR has defined inner structure, which they describe in [18]. To model the organization, we can simplify the structure and use it for the inner organisations of ANS CR. Then, we will generalise the outer organisations – consumers, partners, suppliers, and other stakeholders. Each division is a part of a unit in ANS CR, and we will be using the highlighted boxes in Table 4 as the organizations.

Table 4: Inner organizations of ANS CR

Division	Unit
ATM Division	Operations
Technical Division	
Planning and Development Division	
ATM Training section	**ATM Training and Business Unit**
Business and Marketing Section	
Flight Inspection Section	
Finance section	**Finance and Administration Unit**
Human Resources Section	
Central Logistics Section	
Strategy and Management Support Unit	
Safety Unit	
Office of the Director-General	

The next step is to analyse the services of ANS CR. For most organisations, there is a service that works as a cornerstone for the organisation. In the case of the ANS CR, this service is even in their name; air navigation services. The inner services or outsourced services are then supporting this service. The services create a pyramid with the final service at the top and the consumed services at the bottom. There are services provided inside the organisation in the middle (see Figure 10). As the inner organisations are simplified, the services are too. The management is highly simplified, and some services are entirely missing in our model.

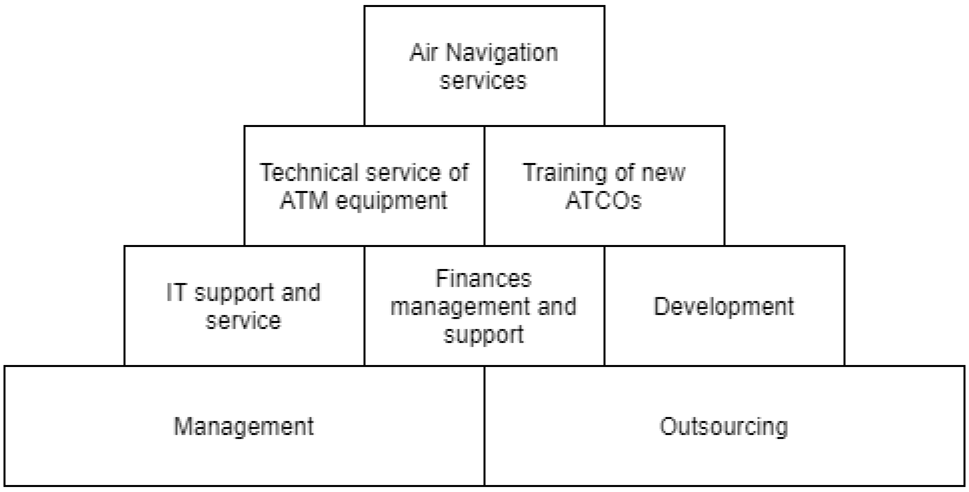

Figure 10: Pyramid of ANS CR services

Finally, we can link the services to the organisations. That will create a diagram in which organisations are as actors inside the ANS CR or outside it (see Figure 11). Each organisation is then a service provider or consumer. The macroscopic connections, like further services of transportation companies or suppliers of construction companies, are missing.

There are many diagrams in [4]. On the other hand, unlike unified modelling language (UML), ITIL is not a graphical language. It does not contain a methodology for modelling an organisation structure nor the services. The model in Figure 11 has three types of elements.

The rectangles are the groups of organisations. That naturally means that it might be the organisation itself. That is the case with ANS CR. The actors, the figures, are the organisations that provide or consume a service. Last but not least, the arrows are the services. Those can be provided either to one of the organisations or a group of them.

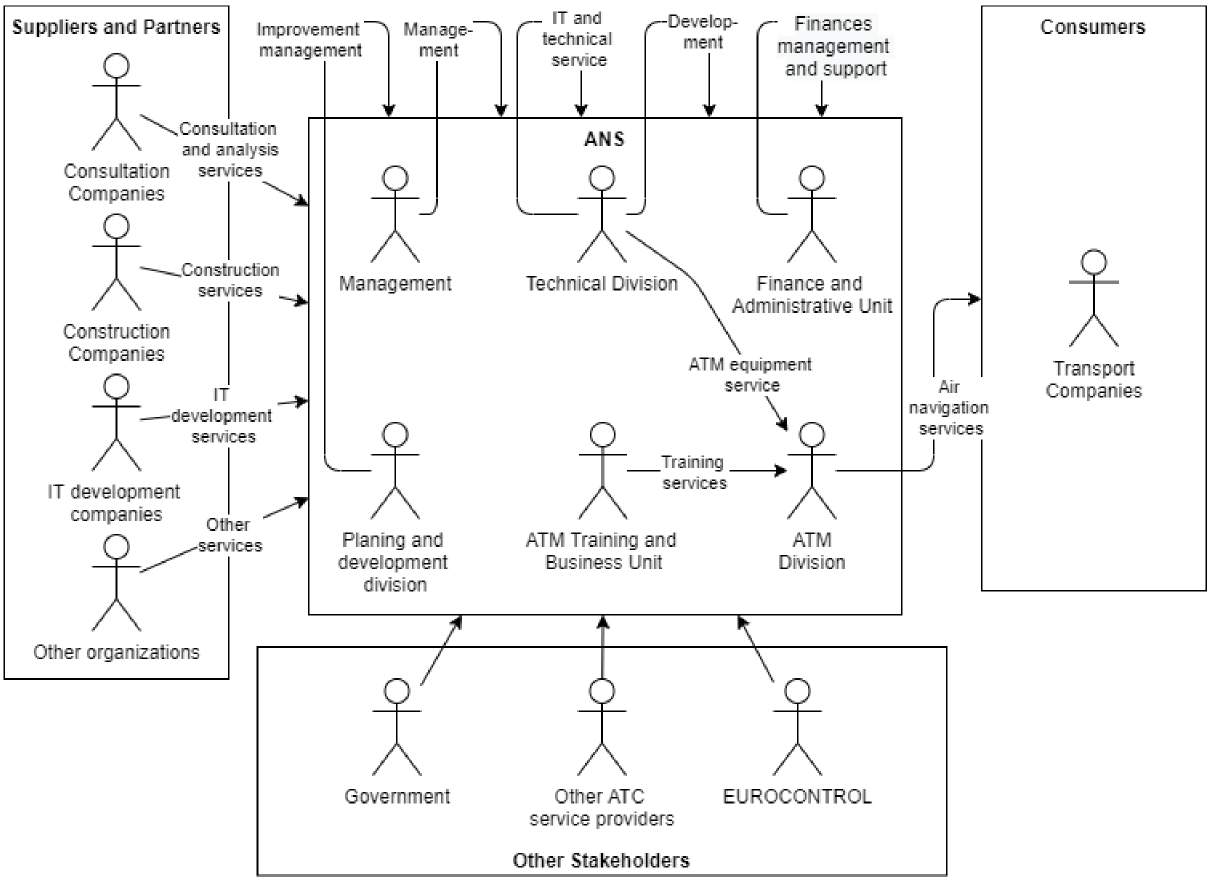

Figure 11: Diagram of ANS CR as an organisation using ITIL methodology

Now, we will look at the technical division a bit more (see Figure 12). Its abbreviation is DTECH, and ANS CR divides it into two parts. The first one is operational, and it is called air traffic management (ATM). The second part is administrative, and its abbreviation is MIS. The MIS composes of servers (PST), technicians (PKT), and development (RIIT).

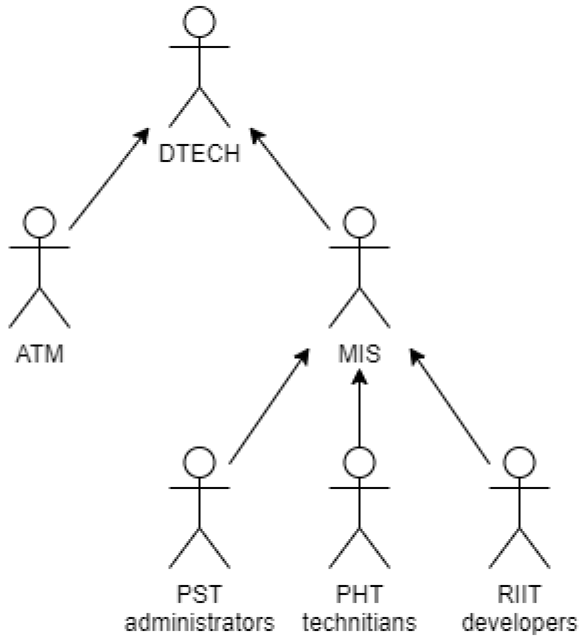

Figure 12: Technical division structure in ANS CR

3.2. Current help desk solution

Organisations should have a way to communicate with users, problem management and asset management. Both the service desk and help desk are IT tools for organisations to communicate with their users. They use a service desk to deliver services to the end-users. A help desk, on the other hand, is a solution for incident management. Nevertheless, in the end, both the service desk and help desk are interchangeable. They are generally two modules of one application, which organisations call various names. [20]

Air navigation services of the Czech Republic (ANS CR) has two help desks. The first one is Alstanet Facility Management (AFM). This software is used mainly for building and physical asset management. However, it also provides a help desk solution and other services depending on the modules used by the organisation. [21]

The second application is BMC Remedy ITSM. A new version of this product is on the market, called BMC Helix ITMS, which replaced the old version. However, it is effectively the same since it offers the same services in addition to others. It aligns with ITIL and offers many services. BMC's customers may choose from modules to have incident and problem management, knowledge management, multi-cloud management, intelligent reporting, change management, asset management configuration management, or digital workspace. [22]

Both the tools are suitable for big corporations and are great for their complexity. On the other hand, ANS CR is looking for a less robust solution. It would also be better for them to have just one help/service desk. Finally, a new single solution for the help desk would save them money. The outsourced systems need experts to set them up. Those specialists are expensive, and their need is, while sporadic, continuous.

Besides those tools, there are three other exciting applications used in ANS CR. The snow (see, e.g. [63]) is used in ANS CR to conduct software asset management audits. It continuously tracks all software on all devices using an agent. The output is a table, and it has an API that makes it possible to get the data for a different system. This system is used only for audits, so it does not connect to anything now.

The Microsoft system centre configuration manager (MS SCCM) is software used for remote installations (see [65]). If an employee asks the technicians to install an application, they can use this system.

We can put all the systems connected with the service desk service into a diagram (see Figure 13). There is another system about which we have not talked yet. It is the technický informační systém (TIS). This system is certified and developed for the active part of ANS CR. This part is under a microscope by the civil aviation authority (UCL) and other organizations. All the software must have a certificate.

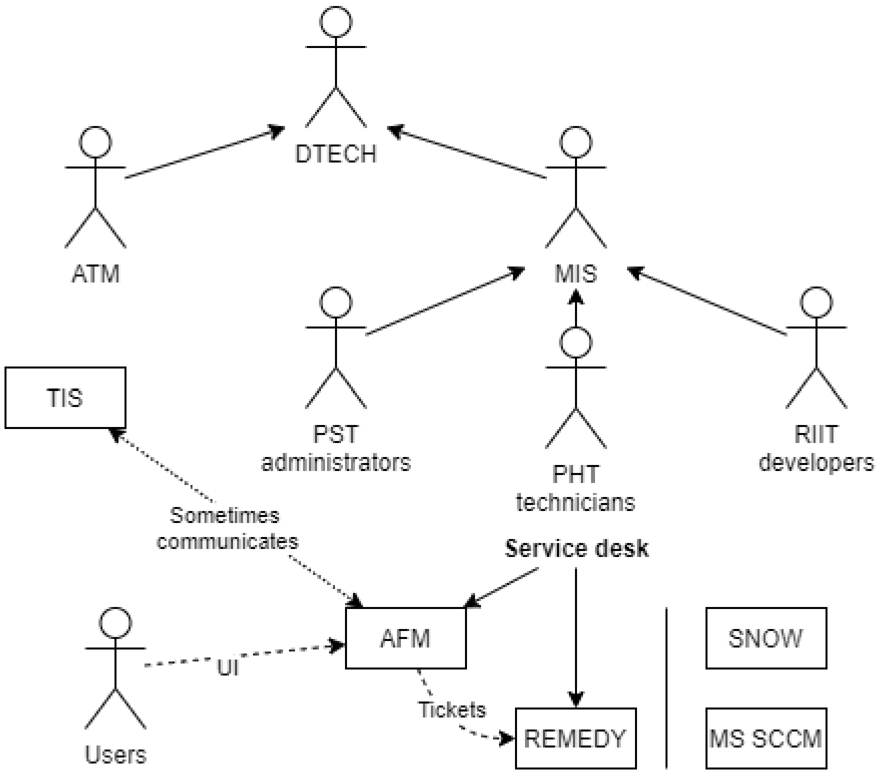

Figure 13: Diagram of current help desk situation in ANS CR

3.2.1. SWOT analysis of the current ANS CR and its help desks

The current solution for the help desk has its strengths and weaknesses. To organise the information, we will perform the SWOT analysis (see Section 2.3.1). We can define the strengths and weaknesses from the description in this chapter. We discussed the opportunities, threats, strengths, and weaknesses during the interviews with ANS CR. We can write all the information into a table to create the SWOT analysis output (see Table 5)

Table 5: SWOT analysis of the current situation

Strengths	**Weaknesses**
A good solution for building management	Expensive
Good software audit solution	Some parts are old and UI inconvenient
Can track software on devices	Robust
Can remotely install software	Difficult to compete
Opportunities	**Threats**
Improve the UI Unite the front end and back end	It becomes even more expensive
Tracking of expected and required software Allow users to use the application on smartphones Connect with other systems	Possible legal problems due to no competition

4. A single help desk for Air navigation services of the Czech Republic

In the last chapter, we have described the current state of Air navigation services of the Czech Republic (ANS CR). That is not a coincidence. Information technology infrastructure library (ITIL) describes how it should be improvements managed (see, e.g. [4]). We have discussed that in Section 2.1.1, but the key message is the seven improvement management stages (see Figure 14).

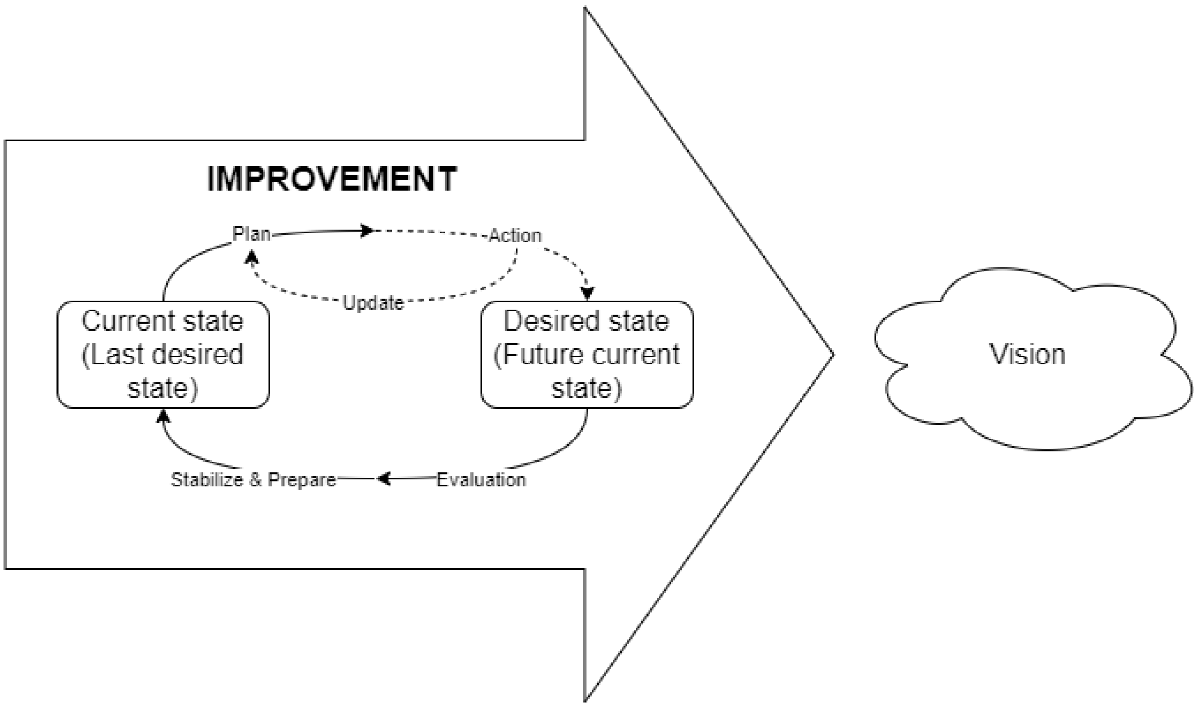
IMPROVEMENT
Plan
Action
Update
Current state (Last desired state)
Desired state (Future current state)
Stabilize & Prepare
Evaluation
Vision

Figure 14: Visualisation of the improvement management stages

The first step is to have a vision. ANS CR gives a general vision in [18]. We have created a vision better suited for the help desk in the introduction. Neither is suitable for proper improvement management because the first one is missing the point and the second has only one.

The second step is to assess the current situation. We did that in Chapter 3. What we have not looked at yet, are help desk solutions on the market. We will look at those in this chapter. The next step is to create a description of the future state. We will be using requirements for that. We can divide those into multiple groups, and there are multiple ways how to gather them. We have dived deeper into this in Section 2.2.2.

The third and fourth steps influence each other. Planning can be done in advance, which we will look at in Chapter 5. However, in the end, some problems will always arise during the action, and we will have to update the plan. After the implementation, the organization must carry out the final two steps. On the other hand, we can plan them out too.

4.1. Market research

The help desk market is extensive. According to [24], there are over 400 help desk solutions on the market. Plus, we are searching for a help desk and a service desk and asset management software.

It seems that Zendesk and Freshdesk are competing for the first place (see, e.g. [23], [25]). On the other hand, SysAid also has a great toolbox and places well. According to [26], in The Czech Republic, there are almost 600 job offers (see, e.g. [26]) that require knowledge of JIRA. It might be because JIRA became a synonym for project management tools. Web Help Desk from SolarWinds is also an exciting solution, thanks to its alternative pricing.

Nevertheless, all help desk solutions listed on [25] are big players. There is one more exciting solution for the help desk. That is MS Office 365. According to [27], organisations with Office 365 with Power Apps can easily make their help desk accessible. The only downside is that the authors provide the solution is straightforward, and it might be hard to connect it to the service desk and asset management.

For the service desk solution comparison, we will look at [28], [29], and [66]. According to [66], the best service desks are SysAid, Freshservice, and SolarWinds. They compare them using their users' satisfaction and market presence. Comparing the first 15 service desks' ranking on all three sites will get similar results (see Table 6). Many systems are joined together with systems for other purposes, like project management, IT asset management, or Help desk.

Table 6: Service desk comparison from different sites

	FinancesOnline	Help Desk Migration	G2	Total rank
Freshservice	1	6	2	1
Zendesk	2	2	–	4
LiveAgent	3	1	–	4
SysAid	4	15	4	5
Samanage	5	–	–	15
ManageEngine ServiceDesk	6	10	15	9
JIRA Service Desk	7	3	3	2
Track-It!	8	–	–	18
BMC Remedy 9	9	–	8	10
Cherwell IT Service Management	10	11	13	11
Agiloft	11	–	–	20
Re:Desk/amaze	12	8	–	13
ServiceNow	13	5	6	6
GoToAssist	14	–	10	18
Spiceworks	15	7	9	9
Saleforce Service Cloud	–	4	–	13
SolarWinds	–	9	1	7
HubSpot Service Hub	–	12	–	22
Help Scout	–	13	–	23
Vision Helpdesk	–	14	–	25
Splashtop SOS	–	–	5	15
Hornbill Service Manager	–	–	7	16
LogMeIn Rescue	–	–	11	20
Ivanti Service Manager	–	–	12	22
atSpoke	–	–	14	25

Finally, we will search the market for an IT asset management system. We will use [67], [68], [69], and [70]. In this case, neither of the sources created a map of the IT asset management systems. We will look at the first 12 systems in each source and calculate their total rank (see Table 7). It seems that the AssetExplorer, Asset Panda, and Pulseway scored the best. According to [68], AssetExplorer is overall the best, while Asset Panda is best for its value. Finally, the Pulseway is the best for its mobile application. The systems provide other services, like service desks or help desks which might influence their market presence and results.

Table 7: Market comparison on IT asset management tools

	Software Testing Help	Investopedia	Capterra	PCMAG	Total rank
ServiceNow ITSM and ITAM	1	–	–	7	6
Nifty	2	–	–	–	10
NinjaRMM	3	–	–	–	12
xAssets	4	–	–	–	15
AssetExplorer	5	1	–	1	1
Atera	6	–	–	–	18
Freshservice	7	–	1	–	6
Spiceworks IT Asset Management	8	–	–	–	21
Snipe-IT	9	–	–	–	24
Asset Panda	10	4	12	3	2
GoCodes	11	6	–	4	6
EZOOfficeInventory	12	–	10	–	24
Solarwinds	–	–	–	6	18
InvGate Assets	–	2	–	10	8
Ivanti IT Asset Management	–	3	–	5	6
Pulseway	–	5	–	2	3
N-cental	–	–	2	–	10
Docusnap	–	–	3	–	12
AssetSonar	–	–	4	–	15
Hexnode UEM	–	–	5	–	16
RMM	–	–	6	–	18
Nlyte DCIM	–	–	7	–	20
SysAid	–	–	8	8	12
UpKeep	–	–	11	–	26
BMC Track-It	–	–	–	9	24
DriveStrike	–	–	9	–	24

We learned that mobile application is a highly valued feature for IT asset management and help desk systems. It is even better if the mobile application can read codes on assets that lead the users somewhere. Familiar interfaces highly improve the customer experience as well as straightforward and easy-to-follow navigation. Software and hardware tracking is also a valuable feature.

4.2. Requirements

We went through requirement gathering in Section 2.2.4. From the possible ways to gather the requirements, we will use interviews and prototypes. Due to the Covid-19 restrictions and hygienic recommendations, it was not possible to conduct group meetings in person. We conducted interviews with specific people in the organization involved in developing and using current and future solutions.

It is impossible to create a functional prototype at the start of the project. We will look at a mock-up version of the product later in the project (see Section 5.5). In the agile project management methodology, the prototypes can be the releases. The team can then track the responses and adjust the requirements very easily.

The ANS CR current documentations inspire the visual style of the requirements (see, e.g. [62]). There, each requirement has its code and description. We will add the source and type of the requirement. The source will indicate from where the requirement came. The type can be functional, non-functional, quality, or constrain (see Section 2.2.4).

Attachment A contains the table of the requirements. It also contains some terms which we have not defined yet. The definition of the terms is in Table 8. We will use the same terms later in the following chapters.

Table 8: Explanation of terms from requirements

Term	**Explanation**
User	User is any person with login information to the system.
Operator	An operator is any user who has higher rights than a user.
Manager	A manager is an operator who can manage tickets or assets.
Technician	A technician is any operator who can solve tickets.

5. Planning

First, we should choose a project management structure. We discussed the different approaches in Section 2.2.1. For the thesis, we will be using a waterfall project management methodology. That is because the requirements are fixed, and we cannot communicate with the customer from this point of view. On the other hand, the development will probably use the agile methodology. According to [37], it might be best for the project to use both. The development team will be close to the customer, and the requirement may change during the project. It is also a small team with a flat organisational structure, and they develop in sprints. The only problem is that they will probably work on other projects simultaneously, which is, according to [37], the biggest problem in agile development.

We will use the agile project management methodology. The requirements presented in Chapter 4 are fixed in the thesis, but they might change during the development. The waterfall methodology is unsuitable due to the development team structure. Agile development will, on the other hand, give the team more flexibility in developing the functionalities. It will also ease the transfer from the current help desks to the new ones.

5.1. Decomposition of the help desk application

The first step in the decomposition is brainstorming, from which we should make a mind map. We shall combine the individual and group brainstorming techniques. The approach is as follows:

1. create the centre point,
2. let the team think of critical functionalities for the current centre point,
3. create a group of functionalities which will become the new centre point,
4. repeat points 2 and 3 until no function groups are left,
5. let the team come up with their groups of functions.

We can see the resulting mind map in Figure 15. The first point was naturally the ANS CR single help desk. The team quickly started with the help desk functionalities, then grouped up into the help desk functionality group. The team named ticket management as one of the points. The team decided to elaborate on that topic more and created a ticket management functionality group. The team then proceeded with the asset management functionality group, which they touched at the help desk. Finally, they added few other ideas. On the other hand, they decided not to elaborate on them further, so they have not become a functionality group.

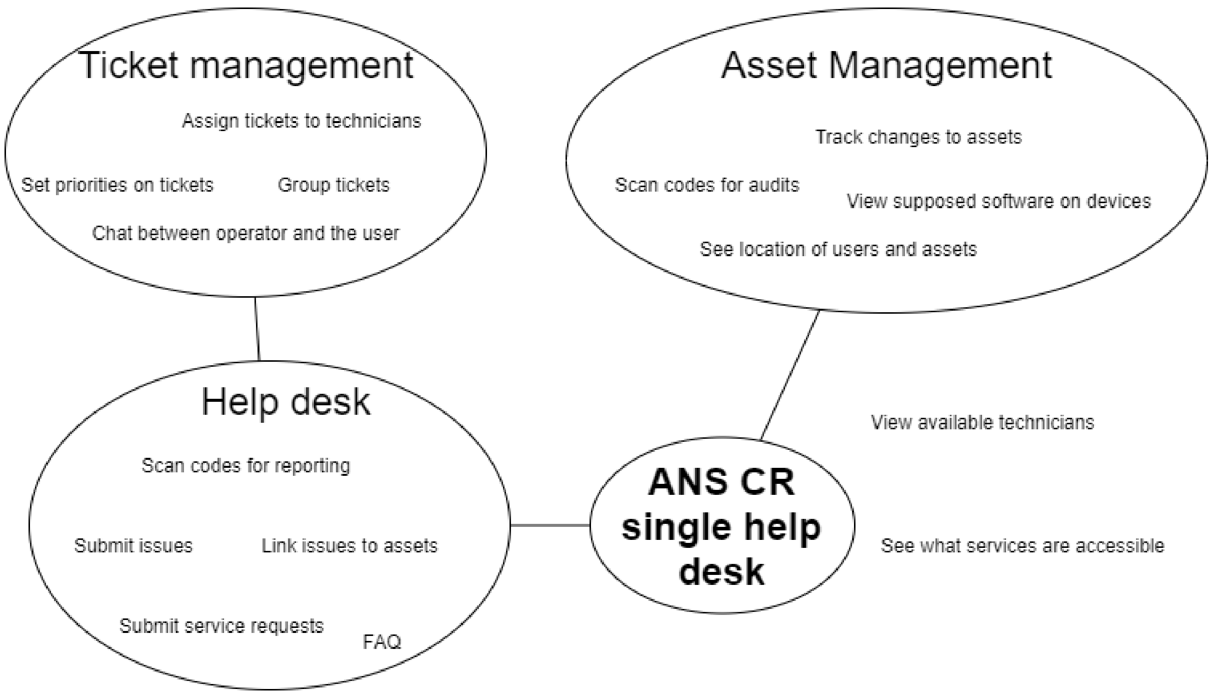
Ticket management
Assign tickets to technicians
Set priorities on tickets
Group tickets
Chat between operator and the user
Asset Management
Track changes to assets
Scan codes for audits
View supposed software on devices
See location of users and assets
Help desk
Scan codes for reporting
Submit issues
Link issues to assets
Submit service requests
FAQ
ANS CR single help desk
View available technicians
See what services are accessible

Figure 15: Mind map from an interview for work breakdown structure

Since we chose the agile project management, the WBS (see Section 2.2.1.1) will have the following structure:

- project,
- releases,
- features, and
- iterations.

We will be using the top-down decomposition. It is good because of its logical approach. Because the requirements might change during the development, new user stories might rise, so bottom-up is not suitable.

First, we will name our project the ANS CR single help desk (see Attachment B). We will divide the project into releases. The zeroth release will be the base app. This module will just allow the users to log in and add new users. The first release will bring communication to the users and help desk operators in a dashboard. The next release will extend the database and the application by implementing IT asset management. The following release shall be the help desk, which will allow the users to create the tickets and manage them. The help desk module joins service requests, incident management practices, and knowledge management (see Section 2.1.3). The last two releases will bring improvements to the customer experience. Finally, the project manager will oversee the project to ensure its smooth course.

5.1.1. Base app

The base app will create a basis for all future modules. This release should focus on the installation package, registration, login, viewing profiles, settings, and logs. We will see that the users cannot do anything in the app yet. The subsequent modules will provide users with new functions.

The **registration** feature will allow the managers to add new users. In this step, we will also need to create the installation. The installation will create the database and write it to the first user. This user can then add new users.

The **login** feature is a security feature, which will forbid access to unwanted users. The users should enter the login information, username and password. The application will allow access if the information is correct. Naturally, if the information is incorrect, the user should get an error message.

The **profile** feature will bring modifications and search to the users. That means that the users will be allowed to see their and others profiles. Thanks to this feature, they will be allowed to change some parts of those profiles. The profile feature will also bring roles to the system. The roles carry permissions with them. Since this WBS includes roles and permissions, it might take longer to finish it. On the other hand, the development team might have some experiences with this WBS, which would speed up the process.

Finally, the **logs** feature is essential for future troubleshooting. Each action that the users could do should have its code. The same goes for the errors. Whenever a user makes an action or gets an error, there should be a new code in the database.

In this release, the users can log in, view and potentially modify profiles, and add new users. All the actions and errors will be monitored and logged. We can see what users can do in Figure 16.

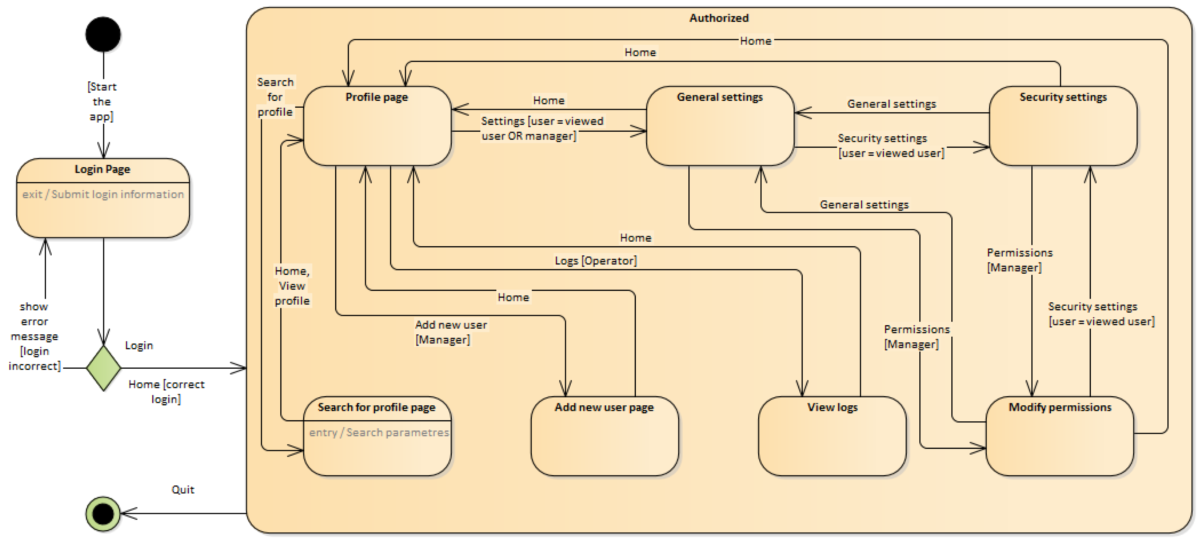

Figure 16: State-Machine diagram of the base app release of the ANS CR single help desk application

5.1.2. Service desk module

The service desk shall be the first module for the ANS CR single help desk application. The service desk management practice directly inspires it (see Section 2.1.3). In this release, the application should be made available for smartphones. The users should also easily access emails and phones from the app, and there should be a new homepage.

The **dashboard** contains many minor functionalities, which would be useless if not together. From now on, the dashboard will be the home screen for the application. There will be announcements about what happened. Since the managers can change the permissions, they will also change permissions to manage the dashboard. There should be three types of permissions for the announcements. First, the viewing permissions will be the basic ones. They will allow users to see the announcements. The creator permissions will allow users to write new announcements and modify their announcements. Finally, the editor permissions will allow the users to create, modify, and delete announcements. They can also change the announcements' position on the dashboard. The announcements themselves should have a name, body, time and date of creation, and author.

The **connection to email and phone** WBS can differ depending on the current requirements. We will model the system just by linking it with a client for such communication. Meaning, when a user views another user's profile and clicks on the email, they should get into the email client. The same link should be in the mobile version to the SMS messages.

The **mobile application** functionality is instead an optimization of current and future releases. The mobile app does not need all the functionalities of the desktop version. The most critical features we have not discussed yet will be creating tickets and asset management audits. In this release, the mobile app should have a login page and a dashboard. It is possible to extend the functionalities in the future updates and as the requirements change, but more on that later (see Chapter 6).

In this release, the users will log in, view profiles, modify some information depending on the permissions. They can view the dashboard and modify it depending on the permissions. While viewing other profiles, they can quickly get to communication channels. The application will log all actions, and operators can view those logs. Figure 17 shows all the pages of the desktop app and how the uses can browse them.

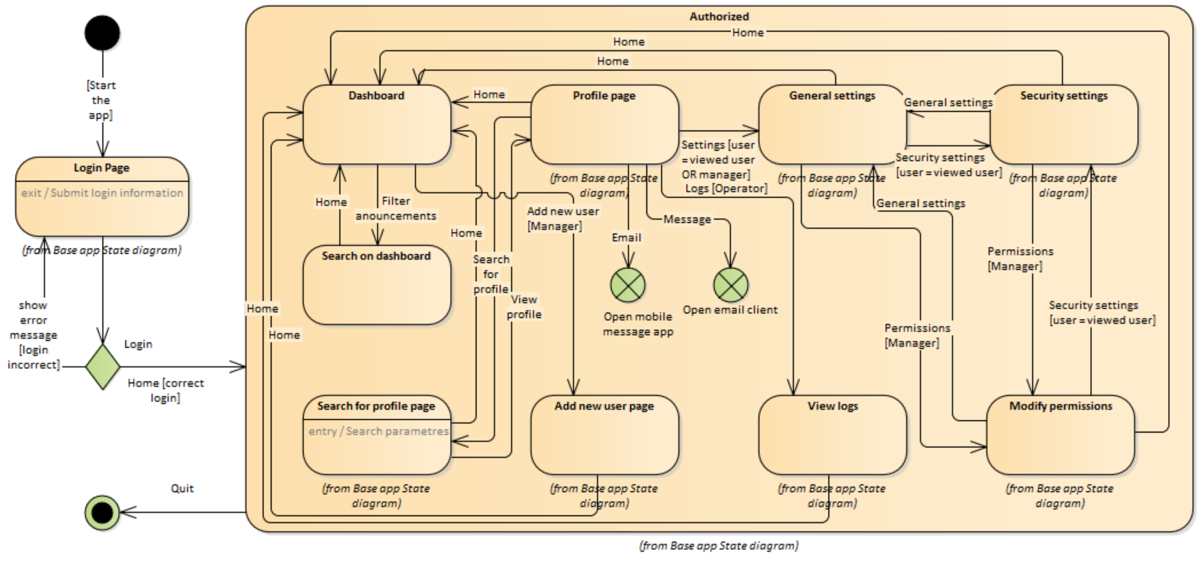

Figure 17: State-Machine diagram of the service desk release of the ANS CR single help desk application

5.1.3. IT asset management module

The IT asset management module will bring the assets to the application. We will use the service management practice of IT asset management (see Section 2.1.3) as an inspiration. It will significantly widen the database, but there will not be many new functionalities for basic users.

The **locations** functionality is a new entity in the database for the locations. It will allow users to add new locations to the database. The managers will connect the users to the locations so other users know where their office is. It should be possible for the locations to have a sort of hierarchy. On the other hand, the managers should create this hierarchy to their image manually. This WBS needs to create a user interface (UI) to create the locations and connect them to the users. Naturally, location management is a question of permissions.

The **assets** functionality will be again a new entity in the database. It will contain all IT assets, which includes hardware and software. Each piece of hardware should have a controller. Though users might share some devices, like printers or projectors, there should be a person responsible for all those assets. It should be possible for the users to add, modify, or delete the assets depending on their permissions. That includes a change of controller or location. The assets functionality includes UI for asset management.

The software assets will be a list of all software assets ANS CR has. Its goal is not to track what applications the users are using but rather to track what software they are supposed to have. The link between the users and software assets will track just that.

The **search** functionality will focus on the communication of the user with the database. The functionality should convert the search criteria to a query that the database will answer. We can create the search engine before and just add new keywords in the IT Asset Management WBS. The UI should contain a basic search, advanced search, and syntax guide. Depending on the permissions, some users will have access only to their assets while others will see all the assets.

The **changes** functionality will allow users to link the assets and locations to contracts in the accounting app. If there are any records of transfer of assets between personnel, the users can link them to the changes. Naturally, only users with proper permissions will be allowed to link the changes.

In this release, we can also extend the **mobile application** by the code reading, which means there should be a code on the assets that the mobile application can read. When the application reads the code, it should direct the user to the asset. This functionality is suitable for audits and, in future, for ticket creation.

Figure 18 shows the application pages and how to navigate through them. To enter some pages, the users will need specific permissions. Usually, the guards on the relationships check the conditions, but we wrote them as entry comments to the pages to save space.

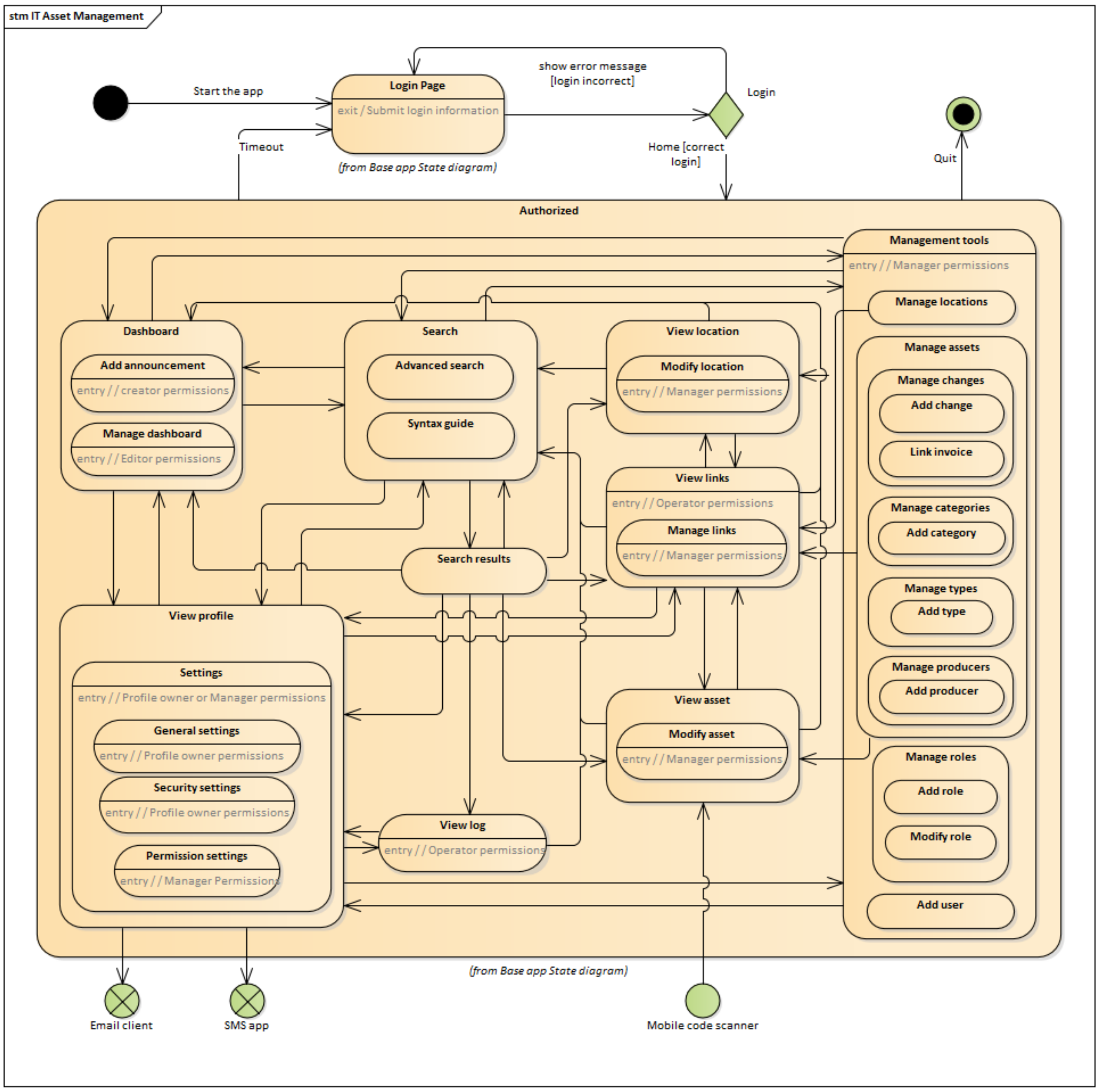

Figure 18: State diagram of pages of IT asset management release of ANS CR single help desk application

5.1.4. Help desk module

The help desk module will allow the users to create tickets. The managers will manage them, and the technicians, or the assigned users, will solve them. Three management practices inspire this module; service request management, incident management, and knowledge management (see Section 2.1.3).

The **tickets** functionality will allow the users to create the tickets. At this point, they will be allowed to report incidents. There will be submission UI for desktop applications and mobile applications. The mobile application can scan the codes on hardware assets to easily access the ticket creation for a better customer experience. All users should be allowed to view tickets created by them and assigned to them. Depending on the permissions, some users should also be allowed to assign and group tickets. Naturally, if a manager assigns the ticket to a user, they should view it, communicate with the author, and change its status.

The **service catalogue** functionality will be a new entity in the database. It will contain information about services the users can request. That will also bring a new type of ticket, the service request. We discussed the difference between the service request and the incident in

Section 2.1.3. When the user scans the code on an asset, they should get to the action choice page from this functionality forward. There should be a link to report an issue, view the asset, and request services. The users can view the services on the service request page for a better customer experience. When a user requests a service, it creates a new ticket.

The **knowledge base** functionality is the FAQ. It should contain all incidents that happened and how they were solved. Users with permissions should care for the knowledge base. That means similar grouping incidents, announcements on the dashboard, and knowledge modification for easier browsing. It should be possible for the users to use the same searching techniques for the knowledge base as in asset management. The knowledge base should be accessible from the mobile applications so the technicians can search for help at the site.

Finally, the **direct message** functionality will allow the operators to leave comments on the tickets. They can use the comments for anything, but it is best to use them to communicate and record actions. The technicians can use different tools to solve the issues. The recording of actions should record the use of those tools. An example of such action is the TeamViewer. If a technician takes over the user and helps them use the TeamViewer or a similar application, the system should log all the technician's actions. That is especially important for the knowledge base.

This release will be fully functional, and the following modules will add new non-essential features. The users can log in, report issues, request services, modify their profile and view the dashboard, knowledge base, user profiles, and assets. Depending on the permissions, the users will be able to modify other people profiles, add new users, manage tickets, solve them, manage the dashboard, search through the assets and locations and modify them. The users can quickly request a service and report issues using the mobile application. The technicians can view the knowledge base using the mobile app.

5.1.5. Availability management module

The availability management is the first improvement to the system. We skip the design of this module because it is beyond the scope of this thesis. On the other hand, it is a nice feature that the development team might want to add. The availability management practice directly inspires it (see Section 2.1.3).

The **schedule for roles** is a feature that should indicate when each person should be at work. It can use data from entries and exits from the buildings to specify the user's current location and check whether they are at work. It is also suitable for the managers to know which technicians they can use for different tickets.

The **vacation** feature will allow the users to quickly request vacation, time off, or any other change in their schedule. They can use it for requests to change shifts. The application can calculate if there are enough workers to take a vacation or someone who could take over the shift. The human resources manager can then view the request and suggestion and allow or deny the request.

5.1.6. Following improvements

The following improvement is suggestions on what features would improve the customer experience. Though it is a single WBS, it has no logical connection. It can be taken apart and analysed further. We will skip the design of this WBS.

The **rules** feature will allow the users to automate some of their tasks. The rules feature should contain conditions and actions. The users can then create rules or scenarios by combining those elements into diagrams. It is possible to create this feature or outsource it (see, e.g. [78], [79]).

Chatbots are a kind of automation that can speed up the solution of the incidents. If some of the issues can be solved at the site by the user, the technicians will have more time for more complicated problems. The chatbots can use the knowledge base and automation.

The **problem detection** will go through the tickets on all assets and correlate them. The problem detection should find assets that need periodical care and might have a bigger problem. (see Problem management in Section 2.1.3)

The **continual improvement register** will allow users to submit their ideas on improvements. The managers can then go through them and use some of them.

5.1.7. ANS CR single help desk project management

The project management WBS is for the design and development project of the ANS CR single help desk. Some might have thought that this WBS is a project management module, but it is not.

Time management is the first part of project management. Typically, we would create a time plan for the project and modify it during the project. That is not applicable here. Since the development will begin after the thesis, the time plan would probably change. The agile project management method just amplifies that.

Risk management should be a continual process during development, similarly to time management. It should monitor known risks and look for new ones. We will talk more about the risks in Section 5.2.

The **release management** should overwatch the releases and implementation of the modules. It should inform all users about new functionalities and teach them if necessary. If there is an implementation management team, they should cooperate with the release management team.

The **requirement management** gathers reports from the release management and updates the requirements. It is essential for the customer experience that the responses from the releases are processed. The requirement management team can use the requirements we presented in Section 4.2 and update them. They should do so at the start of the development as well as during it.

Usually, we would dedicate a single section to the time planning. In this case, we can create a proposal for the time plan (see Attachment D). Rather than time, we can use the percentage of project fulfilment. We propose to re-analyse all the models from the thesis. Then, to develop the solutions and release them. After all releases, the requirement management will modify the requirements, so they fit the current situation. Since the requirements might change, it will be necessary to change the application. The changes should be shorter than the development itself. The changes are short development cycles above the main development line. The system will then enter the operation phase in which the team can still change it.

5.2. Risk for the project

Naturally, there are risks even for this project. Since we are using a combination of waterfall and agile project management methodology, there will be a combination of risks. We looked into general practices of risk management in Section 2.2.2.

The first step is to create a risk management team. According to [55], there should be project management, engineers, employees, health and safety professionals, and maintenance in the team. Unfortunately, the Covid-19 situation changed many things. Along with the restrictions, the air traffic industry took a big hit. That also means that the risk management team reduced significantly. We consulted the risks during the interviews.

We already did most of the next steps in Chapters 4 and 5. The risks can arise from any part of the project. They can be internal or external. In Attachment C, there is a list of risks with an estimation of likelihood and impact. Each risk has a short description, and we proposed possible mitigations. Figure 19 shows all parts of the project in which the risks might arise. Though the bottom four stages of the project are not part of this thesis, they are still part of it.

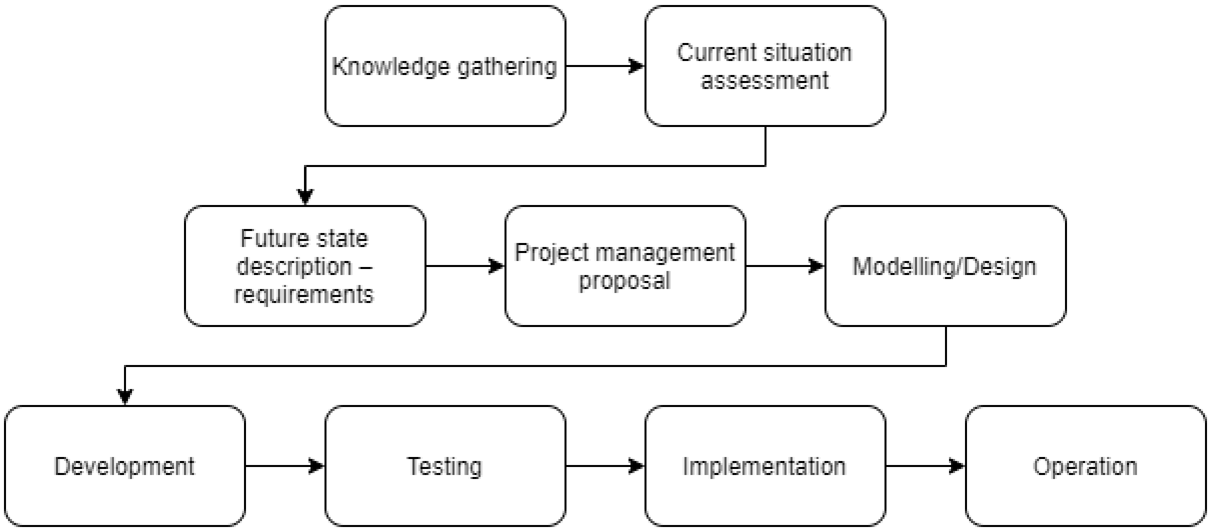
Knowledge gathering
Current situation assessment
Future state description – requirements
Project management proposal
Modelling/Design
Development
Testing
Implementation
Operation

Figure 19: Parts of the project from which risks might arise

5.3. Use cases for the system

In this section, we will use some UML diagrams (see Section 2.3.2). The system use cases will help us define the processes inside the help desk. We will divide the use cases into the same groups as the WBS. We can also utilize the mind map (see Figure 15) and the requirements (see Attachment A). We will use the use case diagram for the first database model and show functions in each class. Finally, we will use the activity diagrams to show the processes in the system.

The actors will be the same for each module. There will be users who will submit the tickets and control the assets. The operators are users too, but they will also add new users and other stuff. The managers will have the permissions to modify permissions and manage stuff. The technicians will have permission to work with the tickets. Both technicians and managers are operators. Though we defined three types of permissions, it will be possible to create roles with combined permissions. That is totally up to the managers. Figure 20 shows the actors of the system. Though the operators should be on the right side of the use case diagrams because they are derived, we will put them on the left because of the generalization relationship between them and the users.

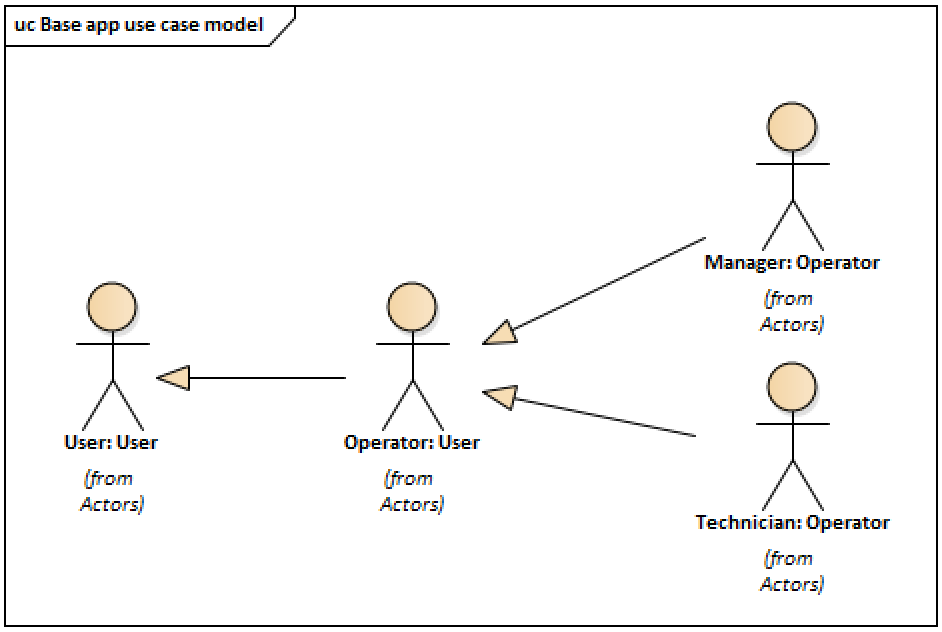

Figure 20: Actors in the ANS CR single help desk

5.3.1. Base app

We already talked about what the base application should do in Section 5.1.1. Figure 21 shows all the use cases and actors in the base application release. We can see that the users will log in to the app and modify their profiles. They can search for other users and add or remove tags on those users. The operators can add new users and view logs. In addition, managers can modify the permissions of other users and install the application. We will look at the use cases closer later in this section.

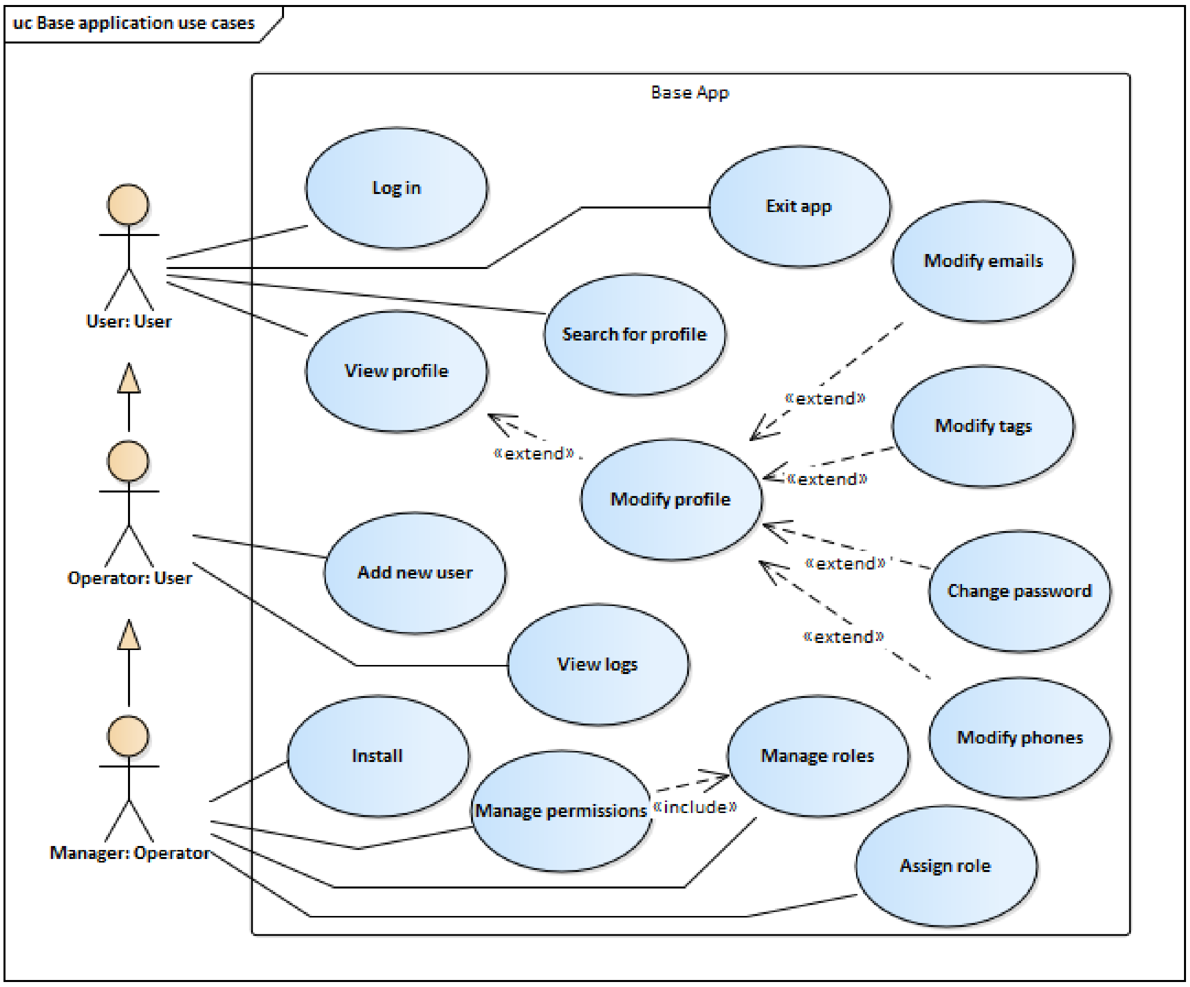

Figure 21: The use case diagram of the base app

In Attachment E, there is a class diagram of the base app. In this release, everything is about the users. That means that the user class is in the middle of everything. The tags are made by and to users. Each user has an email and maybe a phone number.

Furthermore, whenever they do something, the logs will update. We will look closer at the database structure in Section 5.4. There are methods, which we will be using in the activity diagrams (see Attachment G1).

The first use case we will look at is the add **new user**. The activity diagram in Attachment F describes the process of adding a new user to the system. Only operators can add new users. When they click the add new user button, the application will open a new user page. In this window, the manager must fill in the name and surname. They must choose whether to generate a username automatically. That can be accomplished by a pool with the generated username prefilled. The manager can then change it but does not need to. How the application generates the username is defined in the installation use case. Generally, it is a combination of the name and surname with a number if there are any other users of that name. The operator then needs to fill in an email or let the application generate it. The email will be the primary email of the user until they change it. If the manager lets the application generate the email, it will add the predefined domain to the username and save it. If the operator is a manager, they can instantly assign a role to the new user. Otherwise, the new user will get the role of an ordinary user who has the least permissions. The operator must now confirm the data and the wish to add the new user. If the user does, all the input boxes will lock. The operator could cancel the creation till this point by cancelling the creation or exiting the application.

The application now generates a random password and shows it to the operator. They can print this information. Afterwards, the application hashes the password and saves everything into the database and logs the action. If an error occurs, the application will inform the manager about it and provide them with the error code. The error might occur in the communication with the database or any other reason That might help the technicians troubleshoot. If everything went well, the application closes the “add new user” page and opens the home screen again; now, view the profile. The development might decide to create a tool for importing many users, but we are not modelling such a tool here.

The **installation** activity creates the database and then asks the manager to create the first user. The new user will automatically get the highest permissions to add new users and modify theirs’ permissions. During the installation, the user may declare the username generation key and automatic email generation domain. Naturally, the user can cancel the installation at almost any point. It also cancels itself if there is not enough space.

The application will execute the **login** activity anytime the user changes a window. It first checks whether the user already logged in. If they are, the process skips to the end and successfully finishes. Otherwise, the application opens the login page. The user must fill in the username or primary email and the password. Then, they confirm the input, and the application checks it with the database. If the credentials are correct, the application will let the user in and save the information to meet the initial condition next time. Otherwise, they get an error message, and they are not let in. Finally, if the login was successful, the user is let to the home screen. That is either the view profile page or dashboard, depending on the release.

The **search for profile** activity just gets the input parameters from the user, converting them into a query used to search the database. There is no particular page for the search nor syntax guide yet. That will come in the subsequent releases. On the other hand, a special syntax for the search should already exist from this release. The development team might get inspiration from [81], though it is an entirely different area.

The application executes the **view profile** activity whenever the user clicks on any profile button. That is either their profile or another user’s profile. The application gets the user information from the database using the user id from the profile button. The application will then open the profile page and fill it with information about the user. If users view other users' profiles and have proper permissions, they will see the conversation history, assets, logs, and announcements. All users can manage tags they gave to other users. If the user views their profile, or they are managers, they will see the modification button.

The **modify profile** use case extends the view profile. Whenever the user clicks on the modify button, the application executes this activity. It also has many extension points, which lead to different modifications. If the user views other user’s profiles, they will be able to modify tags. Otherwise, or if the user is a manager, they can click on the modification button. The managers can assign a different role to the user. The owner of the profile can modify profile images, passwords, emails, or phones. Naturally, the user can end the changes by saving, cancelling, or exiting the application.

The **modify emails, phones, and tags** use cases are very similar. That is because the user can have multiple of those assigned to them. They can add new ones, change current ones, or delete the current ones. The user can delete all their phones and tags, but they must leave at least one email. There always must be at least one email. All the use cases allow the user to repeat any actions any number of times until the modifications are complete. The **modify**

profile image is similar. If they have none, the user can add a new profile image or change or delete it otherwise.

The **assign role** use case is a bit different to other modifications. The manager can open the permissions settings of the user. There they can click on the role and see available defined roles. The manager might create a new role with new permissions, but that is another use case. The manager then confirms the choice, and the application saves the change into the database. The manager can cancel the change, or there might be an error in the communication with the database.

The **manage permissions** use case is only for the managers. They can enter the role management page. There they can **manage roles**. Meaning that they can create, modify, or delete roles. To create a new role, they need to click the new role button and manage their permissions. The “modify roles” are similar because they click to modify the role button and then manage its permissions. The managers can delete roles only if no user is assigned the role. The manage permissions page is a list of all activities and permissions for the role. A role has all the permissions automatically at the minimum level. The manager must change them to allow the users to do something. The manager then must confirm the changes, and the application will save the new or changed role into the database. The role-based access control inspires this process (see, e.g. [80], [82]).

Finally, the **view logs** use case is accessible for the operators, but this might change since the managers can manage roles. All the use cases have the update logs at the end of their activity diagrams. It saves the logs into the database. If the operator wishes to see the logs, they click on the view logs, either in the menu, hot bar or while viewing a user. Depending on where they entered, the logs might filter instantly. For example, if they entered from the user page, they will see all the logs belonging to the user. That will apply even for future updates. The operators can then change the filters. It is possible to do that using the same syntax which we defined for the user search. The syntax should include a way to sort the items.

We can model the activity diagrams to describe the use cases further. Attachment G1 contains Those activity diagrams. There are more activity diagrams than use cases because the diagrams would be confusing. We will need this attachment to open the following Attachments G.

5.3.2. Service desk

We described our expectations for the service desk in Section 5.1.2. Figure 22 shows all the use cases for which the users can use the application. The mobile application has no use cases because it shares all the use cases of the current system. We might exclude some use cases to the mobile app, but that is up to the development team.

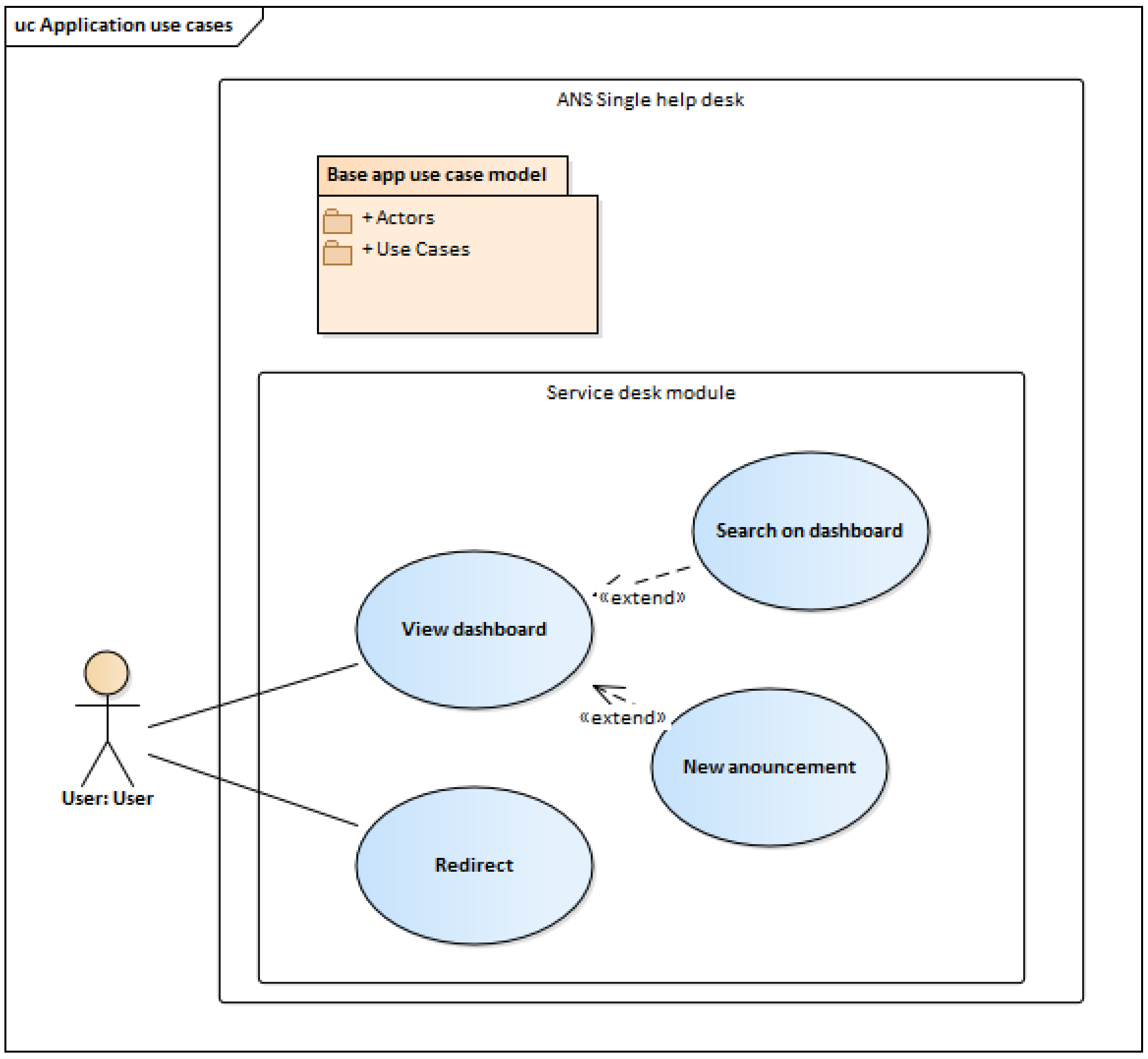

Figure 22: Use case diagram of the service desk release of the ANS CR single help desk

The connection to the email and phone does not require any new entities in the database. The class diagram of the service desk (see Attachment E) has only one new class compared to the base app. It connects to the user and the tag classes. That is because each announcement must have an author, and the author can tag each announcement. Though it is not necessary to tag the announcements, it is better for searching.

The dashboard is the home screen of the app since this release. After the login and whenever the user clicks on the home button, they will execute the **view dashboard** use case. The application will open the dashboard page and fill it with the announcements with the highest priority. The age of the announcements determines the priority. The newest announcements have the highest priority. On the other hand, those users with permission to manage the dashboard can change the priorities of the announcements. The user can click on the author to view their profile or search all announcements with this tag.

We already touched the **search on the dashboard** by the user clicking a tag on an announcement. The search on the dashboard starts with the user clicking the search button on the dashboard screen. The application will open the search on the dashboard page, where the user can enter the search parameters. It can again use the same syntax as the user and log search. The application converts the parameters into a query and searches the database.

The user can modify the sorting of the results, but it will sort from the newest to the oldest if they did not.

The user can add **new announcements** if they have proper permissions. If they do, they must click the add announcement button on the dashboard. The application then shows them the add announcement page. There they can fill in the name, body, and tags. They can also modify the time of release. The announcements can use HTML for formatting and linking to other pages. The user then confirms the publication of the announcement, and the application saves it into the database. When the time comes, the application will publish the announcement on the dashboard.

The user will initiate the **redirect** use case by clicking on an email or phone number. That generally happens from the view profile page, but the requirements might change during the development. Whenever the application registers a click on an email, it opens the user's primary email client, automatically creates a new email and adds it to the recipients. The exact process will happen for the phone number only on the mobile app. It will instead open the SMS client and fill in the phone number instead of the email.

Attachment G2 contains all the use cases converted into the activity diagrams. The description we presented here is derived from the activity diagrams. That also means they are more detailed, though they might be divided into multiple diagrams. We need Attachment G1 to see all the elements of the model. We will encounter the same problem with other Attachments G because they build on each other, which means that the third release needs all previous versions. The fourth release will ned all three previous versions.

5.3.3. IT asset management

We described the IT asset management module WBS in Section 5.1.3. We will use this and the requirements to create the use cases. Figure 23 shows all the use cases. The whole system contains all previous use cases using the packages.

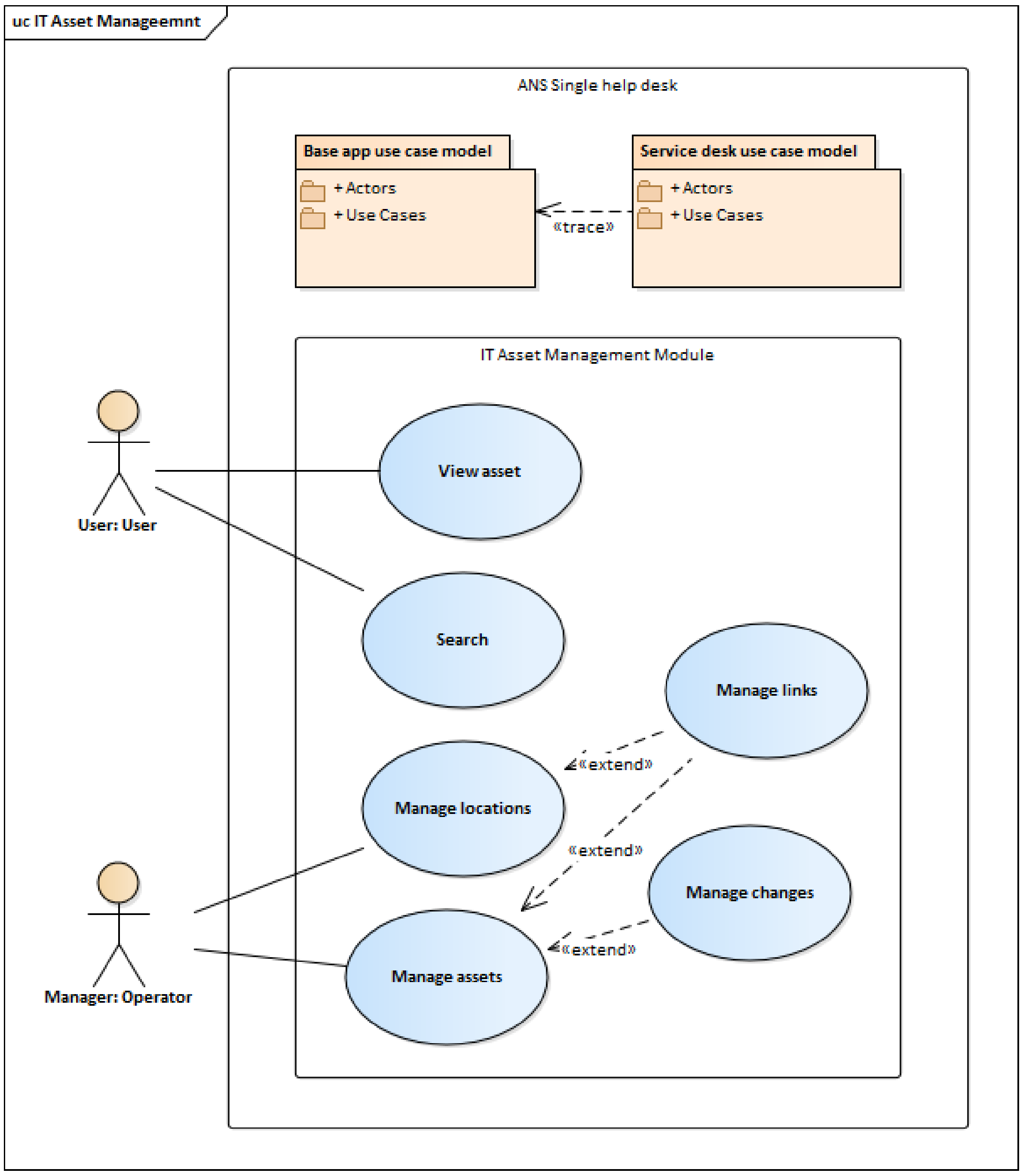

Figure 23: Use case diagram of IT asset management module of the ANS CR single help desk application

The class diagram of the IT asset management release of the application grew by the locations, tickets, and changes (see Attachment E). There are, in total, nine new classes with methods. Those classes are connected only to each other and the user. They do not use any other previous class.

The **manage locations** use case is the first we will go through because it relates to WBS locations. The managers can open the management page, where till now they could have found only the roles. Now, they will be able to open asset management and location management. If the manager clicks on the location management button, the application will open the location management page. The application will load all the existing locations. The manager can filter through the locations, add new ones, modify them, or delete them.

The manager filters through the locations using the data about them. The manage locations page should have a search included with the possibility to open an advanced search. We will talk about that use case in a moment.

The manager can add new locations by clicking the add location button. The application then opens the new location window. The application will allow the user to fill in id, name, and GPS coordinates. The manager may then choose to group the locations. [34] describes the usefulness of the grouping and how to build the hierarchical structure. The manager then confirms the creation, and the application saves the data into the database. The manager may cancel the creation at any time. If an error occurs, the manager will be informed about it, and the application will show them the error code. The application logs any result of the process.

The manager can delete or modify the locations on the manage locations page. There, they can click on the appropriate button. If they chose to delete the location, the application would ask for confirmation of the removal. If the manager confirms the change, the application will delete the location from the database. If any locations are in a group under this location, the application will remove them too. The application will ask the manager if they want to remove the linked asset and users. The application cannot remove the users unless the manager removes them directly. If the manager chose to modify the location, the application would unlock the data for the manager. Then, they can modify id, name, GPS, or grouping location.

The managers can **link locations to assets and users** in two different possible ways. The managers can either create the link from both sides. To link from the location side, the manager must open the location management page. There, the manager can click on the links button. The application will open the links page and show which users, assets, and locations are linked. The manager can add links, modify them, or delete them.

To add a link, the manager must click on add link. The application will open the add link window. There, the manager must choose what the link will be; a user or an asset. To link the location to another, the manager must use modifications in the manage locations page. The link between locations creates a group. After the manager chooses which entity they want to link the location to, they must choose a specific entity instance. They need to search for the user or asset and choose it. When the manager finishes, the application asks them to confirm the creation. If they do, the application will save the link to the database and update the links window. The manager can stop the process at any time. An error might occur during the process, and in such a case, the application shows an error message with the appropriate error code to the manager.

The manager can modify or delete a link by clicking on a specific link. The application will let the user choose to modify the link or delete it. To delete the link, they must confirm the removal when they choose it, and the application deletes it from the database. If they choose to modify the link, the application asks them if they want to change the location or the target. In both cases, the manager must choose a new location or target for the link. The target must be of the same type. That means that if the link targeted a user, it must target a user even after the modification. When the manager finishes the modifications, they click on done, and the application asks them to confirm the modification. If they do, the application modifies the database accordingly. The manager can click elsewhere any time and cancel the changes and close the choice window.

To link the location to an asset or a user from the target’s side, the manager must open a specific user or asset. There, they can click on the modification button if they have proper permissions. The application will open the modifications page, and the manager can change the location. The application will ask them to confirm the changes. If the manager does, the application saves them into the database. The manager can cancel the modification at any time.

When the manager opens the management page, they can click on **manage assets**. If they do, the application will open the manage assets page and show all assets. The manager can filter through the assets using the search tool, which we will talk about shortly. The manager can also open asset category, type, or producer management on the asset management page. They can add, modify, or delete the categories, types, or producers in those pages.

To add assets, the manager must click on add asset button. The application will open the new asset window. There, the manager must fill in the name, date of acquisition, and serial number. The manager must choose a category, type, producer, and controller. The manager then may choose the location, add GPS, and change id. Then, they must click on add button and confirm the information. If they confirmed the creation, the application saves the data into the database. The manager can cancel the process any time by clicking the cancel button or exiting the application. The process might also stop if an error occurs. If it does, the application will show the appropriate error message to the manager.

If they want to manage a specific asset, they hover over it with a cursor. The manager can then click to modify or delete the asset. If they choose to delete the asset, the application will ask them to confirm the removal. If they do, the application removes the asset from the database. Suppose they choose to modify the asset. All information unlocks, and they can change them. The actions then can or must do the same as in adding a new asset. When they finish, they must confirm the modifications, and the application saves the changes to the database. It can be cancelled the same way as the creation of a new asset. The same is true for the error and error messages.

The **manage changes** use case is for recording acquisition, disposal, or any other change of an asset. There are two possible ways how to record a change. The first one is to link an invoice from an accounting application. The other is to create the change manually. To record a change, the manager must search for a specific asset or a group of assets. When the manager chooses all affected assets with the change, they must click on record change. The manager then chooses the type of change, and if there are any recipients of the change, they need to add. The manager can link an invoice to the change at this point. The manager then must confirm the creation and the application saves the change into the database. The process can be cancelled at any point the same way as in previous use cases. Managers may modify or delete the changes by clicking on the change they wish to change by searching specific assets. They can manage the change types and all changes from the management page.

The **search** use case finally allows users to enter the advanced search page and syntax guide page. The users can enter any parameters into the search line. The application converts the search parameters into a query and returns all matching data from the database. Users can click on the advanced search button and syntax guide button next to the search line. If the user clicks on the syntax guide, the application will show a page explaining the syntax. The advanced search button leads to the advanced search window. The application allows the user to specify any parameter. There will be the same parameters the user can enter the search using the syntax.

The **view asset** use case is for viewing a specific asset. The users can view all their assets while viewing their profiles. The operators can even see other users' assets. To view the specifications of an asset, a user must click on it. The application will show them an asset window that will fill with data about it. Naturally, the higher the permissions, the more the users will see. Managers can click on modifications and manage changes to the asset.

We can convert the description of the use cases in this section into activity diagrams. Attachment G3 contains the IT asset management activity diagrams. There are more diagrams than use cases because it is better to create more clear diagrams than one big but confusing one.

5.3.4. Help desk

The help desk is the last update we will model. We described its WBS in Section 5.1.4. We said that the help desk module should allow users to report issues and request services. The operators should be able to manage the tickets and solve them. The module should also contain the knowledge base. Figure 24 shows all the use cases for the release. We will describe the use cases more later in this section.

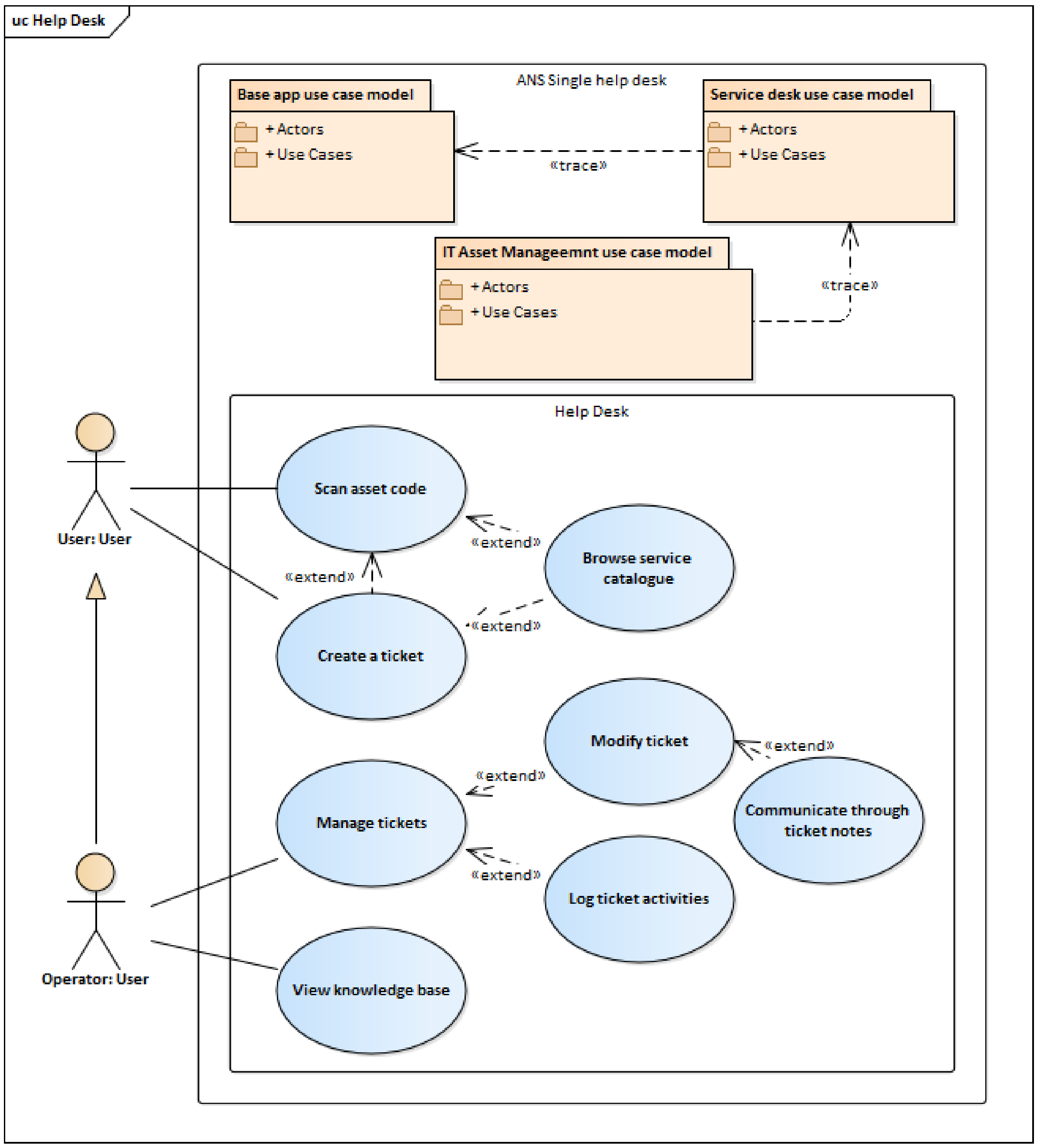

Figure 24: Use case diagram of the help desk release of the ANS CR single help desk

This module will need to store data about the tickets. The tickets can be either service requests or incident reports. The service requests are requests of some services in the service catalogue. The users can request service on an asset. A ticket points on an asset, and there is always one author of a ticket. The ticket should always have a solver. The users use messages to communicate to solve the ticket. Because the ticket will become part of a knowledge base after its solution, the database must store all performed activities. The help desk class diagram in Attachment E shows all the new classes and connections between them. It also contains all methods which we will use in the activity diagrams.

To **create a ticket,** the user can click either the report issue button or the request service button. The application opens the new issue page if the user clicks on the “Report issue” button. The user can fill in the issue's name, description, and asset on the left side of the page.

On the right side, there are the user’s previous reports. The user can click on one of those reports, and the application automatically fills in the pools with the same data as that ticket. If the user clicked on the “request a service” button, the application would open the service catalogue. The user can **browse the service catalogue** on the left side of the page. On the right side, they can see their request history and copy one of the requests. After the user filled in the ticket, they must submit the ticket by clicking the submit button. The application automatically adds id, status, priority, author, and solver to the ticket. The status is zero, priority is high – 3, and the solver is a manager with the least tasks. The database saves the data. The user can cancel the process at any time. It can also end with an error, in which case, the application informs the user which error happed.

The user can **scan asset code** using the mobile device. After scanning, the application opens a window with available services for the asset, the report issue button, and the see detail button. The user can easily report an issue or request a service by clicking a button on the screen. The application opens a ticket creation page with some prefilled data, thanks to the scanning.

Users can view their tickets in their profiles. If they click any of the tickets, the application allows them to add comments to it. The user can write a message and then send it. This way, the author and the solver can **communicate through ticket notes**. The application saves the message to the database. If the user has permission to modify other people tickets, they can see the tickets made by different users. In addition, the operators have access to the ticket management page. There they can see all tickets and all tickets assigned to them. They can modify or delete them. If the operator chooses to delete the ticket, the application asks them to confirm the choice. If the user confirms it, the application deletes the ticket from the database. If the user chooses to **modify the ticket,** the application opens a detail of the ticket. The user can change name, description, asset, solver, status, or priority. They can also add a message or log activity. After the operator finishes the modifications, they must confirm them. If they do, the application saves the changes to the database.

The application **logs the ticket activities** automatically unless the action is not cloud-based. If the technician performed an activity in real life, then they must log the activity. They do that in the ticket modification. They can choose from existing activities, but if the performed activity is not standard, the technician must shortly describe it.

If the technician changed the ticket status to “solved”, they solved the ticket. That will prevent the ticket from appearing in the ticket sections. On the other hand, they can now see it in the knowledge base thanks to the **view knowledge base** use case. By solving a ticket, the technician expanded the knowledge base. The technician can filter through the solved tickets to find one they are solving now and see possible solutions immediately.

All the activity diagrams connected with the help desk are in Attachment G4. We described them in words in this section, but they are in greater detail as activity diagrams. We cannot thoroughly browse the model unless we have all previous versions of Attachment G. We will need the whole Attachment G to see the model correctly.

5.4. Database

At this point, we defined all the use cases and how will the application perform the activities. The class diagrams give us an idea of how to build the database. On the other hand, a class diagram is not a data modelling tool. That means that we should create an entity-relationship diagram (ERD) (see Section 2.3.3). There are many tools for creating the ERD and the SQL script to create the database. In this thesis, we will use apricot (see [84]).

Attachment F contains the ERD of the ANS CR single help desk, and Attachment J contains the SQL CREATE code for the database. In this section, we will look closer at the entities and relationships. We will divide the ERD into multiple sections connected to other groups only in few points, generally one. We skipped the modelling database for each release.

5.4.1. User

The centre of everything is the user entity. Its primary key is the ID, which is an integer. It uses auto-increment. Another unique attribute is the username. It is not used as a primary key because it is a string.

On the other hand, the ID is a number that makes it easier to use in other entities as a foreign key. The username's type is char with a length of 8. The user also has a name and surname attributes, each char with a length of 16. The users also need a password. The length of the hashed version might differ, but hopefully, 64 characters will be enough. The role and room attributes are foreign keys from other entities, but they are integers for keeping the data as small as possible. The room is the only attribute that might have been null in its value. (see Figure 25)

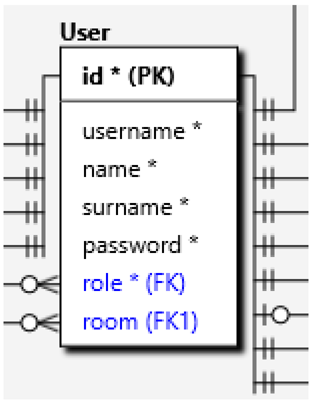

Figure 25: The user entity

5.4.2. The email–phone area

The email phone area is just entities that add more attributes to the user. Since the user might have more emails and phone numbers than one, there must be an entity for each of those.

The email has its owner, name, domain, and priority. Though the combination of name and domain must be unique, the entity should have an ID. Furthermore, since there are duplicate domains, we will create a single entity for the domains too. The ID if the email is an integer and the user, domain, and priority. The domain and user are integers because they represent ID in a different entity, while the priority is an integer because it is easy to represent it by numbers. The name of the email will naturally be a character type of length of 16. There is no attribute in the email entity which could have been null in its value. The email domain entity has an ID, name, suffix, and generation-key attributes. The name and suffix are characters of length 16 and 8, respectively. The generation-key attribute has a "tinyint" type, and it points out which domain it should use during the generation of emails for new users. Again, none of the attributes can have null in its value. (see Figure 26)

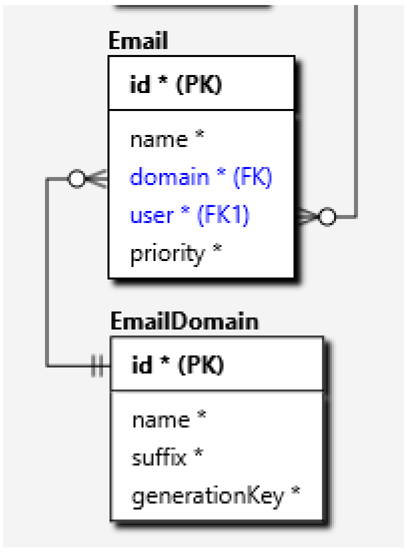

Figure 26: The email and email domain entities

The phone entity composes of ID, code, number, user, and priority. The code is the prefix of the phone numbers, which is the same for multiple users. Therefore, we will create a single entity for the phone codes. The phone code entity has three attributes; ID, code, and number length. All of those are integers. The ID is the primary key, the code is the prefix without the plus sign, and the number length is the number of digits in a phone number with the code. For example, a phone number with Czech code 420 has a standard length equal to 9. The ID, code, user, and priority are all integer types, while the number is a character type with a length of 13. That is because the integer type forgets the last positions of the long numbers, which the phone number are. (see Figure 27)

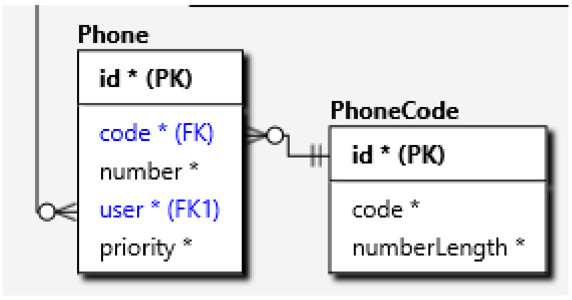

Figure 27: The phone and phone code entities

5.4.3. The tag – announcement area

The tag announcement area connects the tag entity with the user. There are two ways how users can use the tags on other users and announcements. The tags have ID and name, where ID is an integer, and the name is character type with a length of 16. The ID is the primary key though the name is unique too.

The announcements have ID, name, body, author, and time of creation. The ID and the author are integers, while the name and body are character types. The name can have up to 64 characters, while the body may have up to 1024. The time of creation has the type of DateTime. The ID is naturally the primary key, and the author is a foreign key from the user entity. The announcement can have multiple tags, and one tag can be on multiple announcements. Therefore, there is an entity joining those two. It has just ID, tag, and announcement attributes. All of them are integers. The ID is the primary key, and the other two are the foreign keys from the other tables. (see Figure 28)

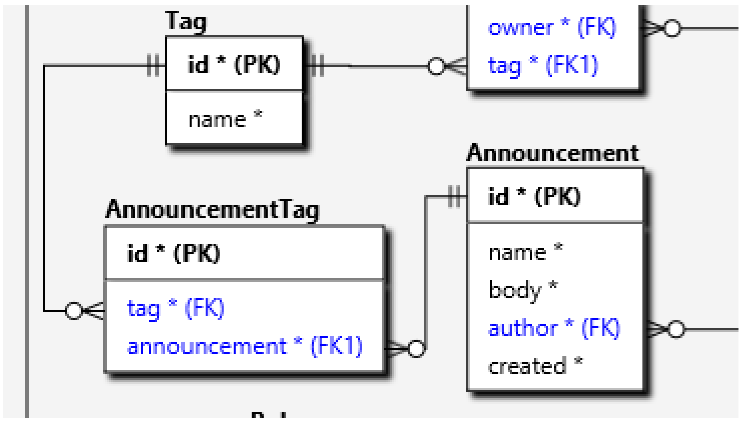

Figure 28: The tag and announcement entities

If a user tagged another user, the application saves in the UserTagOwner entity. This entity has an ID, owner, and tag attributes. It does not have a target because one user might have tagged multiple other users with one tag. That means there is another entity called UserTagTarget, which has ID, target, and tag attributes. The tag in this entity refers to the first table rather than the original tag entity. (see Figure 29)

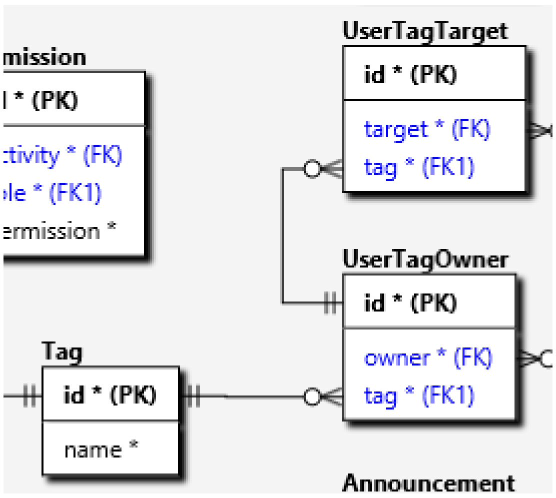

Figure 29: The entities for tagging users

5.4.4. The log – permission area

The log – permission area, connects the activity entity with the user entity. The activity has ID, name, and description. The ID is the primary key, and it is an integer. The name and description should be unique, but their type is char of 16 and 256, respectively. That makes them much more space-consuming and therefore unsuitable for primary keys.

All the activities of users are logged, which makes the first connection to the user entity. On the other hand, logs also record errors. The errors are a different entity, which is the same as activity except for the source. That is a foreign key from the activity entity. The connection exists because almost all errors cohere with activity. The log entity has ID, user, activity, error, and time attributes. The time has the DateTime type, while everything else is an integer. The ID is the primary key, and all other integer attributes are foreign keys. Since the errors and activities are not in a single entity, there is a connection between all three tables. The activity and error attributes can have a null value. (see Figure 30)

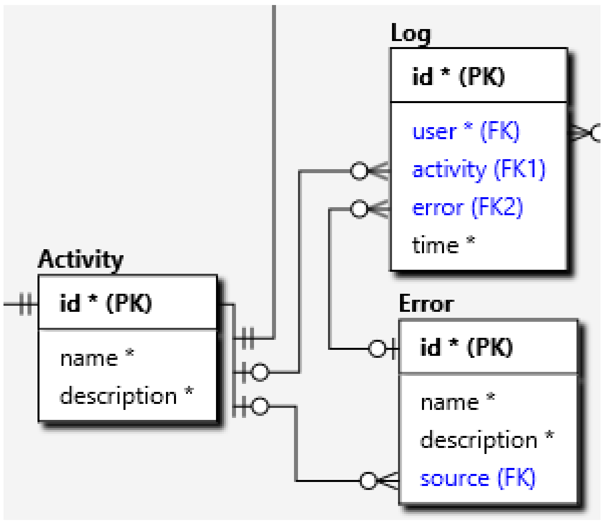

Figure 30: The activity, error, and log entities

The second connection is the permissions. Since we are using the role-based access control, there are entities for permissions and roles. The role is defined only by the ID and name. Both of which must be unique. The ID is an integer, and the name is char type. The ID is the primary key, and each user must have a role. The role needs permissions for activities. The permission entity contains each activity for each role, and each combination has a specific permission. It has an ID, which is the primary key. All the attributes are integers. The activity and role attributes are integers because they are foreign keys that refer to IDs. It is possible to convert the permissions to integers. (see Figure 31)

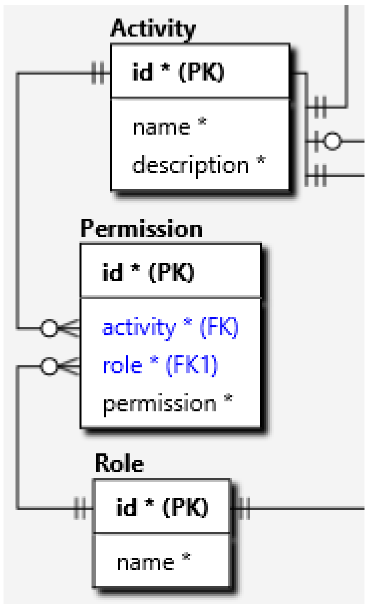

Figure 31: The activity, permission, and role entities

5.4.5. The ticket–asset area

The ticket–asset area, includes most entities. The activity entity from the previous area (see Section 5.4.4) and the user (see Section 5.4.1) connect to this area. It contains the necessary entities for IT asset management and helps desk modules for the application.

We will first look at the IT asset management part of this area. The asset is the primary entity. It has the following attributes: ID, name, category, type, producer, controller, location, GPS, acquisition date, and serial number. The ID is the primary key, and the foreign keys are category, type, producer, controller, and location. The first of the mentioned have their small entities, which each contains just an ID and name. The location is more complex than the others since it can be grouped and has its GPS. The controller is a reference to the user entity. The name of the asset is a character type of length 16. The GPS is coordinates converted to the decimal type with 15 numbers total, 10 of which are after the decimal point. The GPS in the location entity is the same. The acquisition time in the asset entity has a DateTime type, and the serial number is 16 characters long char. There is the AssetChange entity to store data about the changes of assets. Since one change can direct multiple assets, there is another entity called ChagedAsset. It contains just ID, asset, and change, all of which are integers. The ID is the primary key, while the other attributes are references to the asset entity and the AssetChange entity. The AssetChange entity contains ID, type, author, and recipient. All of those are integers because they are keys. The ID is primary, while others are foreign. The author and recipient refer to the user entity, and the type refers to a new entity. This entity contains just a name and ID. The ID is the primary integer key, while the name is 16 characters long char. Finally, there is the Invoice entity, which contains links to invoices in the accounting application. Besides the link attribute, it has a change and ID. The ID is the primary key, while the change is a foreign key referring to the AssetChange entity. (see Figure 32)

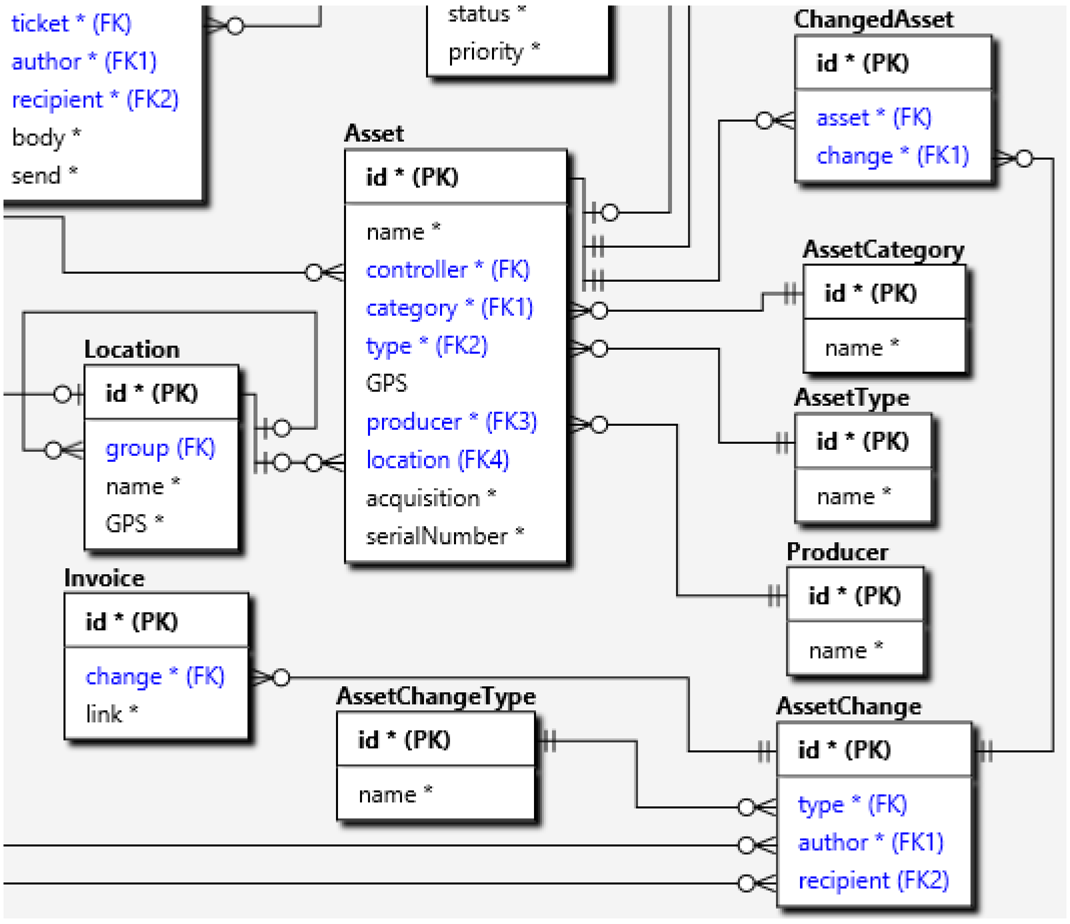

Figure 32: The asset and location entities with related tables

The help desk module will bring the tickets. They have a single entity in the database called Ticket. Its attributes are ID, name, description, time of creation, author, solver, chosen service from service catalogue, related asset, status, and priority. The ID is the primary key, and it is an integer. The author and solver are user IDs from the User entity (see Section 5.4.1). The name and description are both char types of lengths 32 and 256, respectively. The time of creation has DateTime type, and the status and priority are integers. The chosen service from the service catalogue is another integer foreign key. It is from the service catalogue, which will be the list of available services. Its attributes are ID, name, and description. The ID's type is an integer, the name's type is char with a length of 16, and the description's type is char with a length of 256. The services connect to the assets. However, there can be multiple services provided to one asset and one service provided to multiple assets. Therefore, there must be another entity connecting those tables. Its attributes are ID, service, and asset. The ID is the primary key, and the others are foreign keys referencing the service catalogue and asset entities. The solver and author might use the messages, which is another entity in the database. Its attributes are ID, ticket, author, recipient, body, and time of creation. The ID is the primary key, while a ticket, author, and recipient are foreign keys.

All of those are integers. The ticket connects this entity with the ticket, and the author and recipient connect it with the user. The body has a char type with a length of 128. The time of creation has a DateTime data type. Finally, the activity in the ticket entity is for recording what activities the solver performed during the solution of the ticket. Its attributes are ID, activity, ticket, time, user, and description. The ID is the primary key, and its type is an integer. The activity, ticket, and user are foreign keys that refer to the activity entity, ticket entity, and user entity. The time attribute has DateTime type, and the description has char type with a length of 256. The description might have a null value. Generally, the choice of the activity is described enough in the activity entity. If the activity is described generally in the activity entity, the solver should describe it further. (see Figure 33)

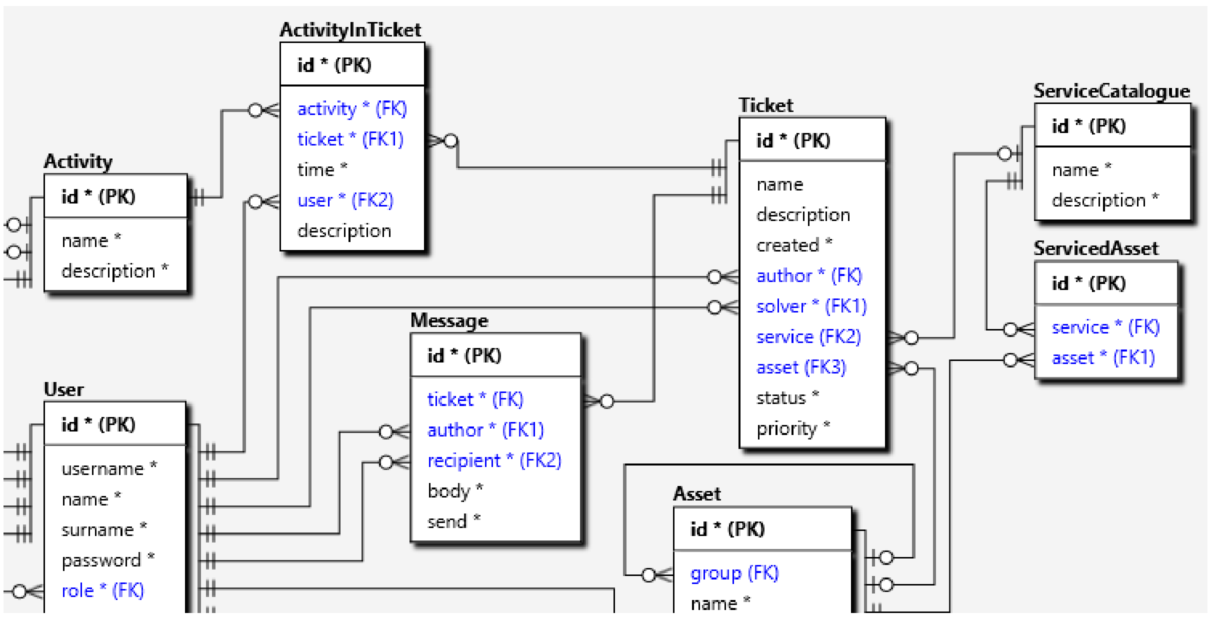

Figure 33: The ticket entity with related tables

5.5. Visualization

Now, we described all the processes in the application and the database. We presented few state diagrams in the description of the first four WBS (see Figures 16, 17, or 18). We also talked about prototyping requirement gathering (see Section 2.2.4). We can create a visualization of the current design. The customer might come up with updated requirements, which might improve the customer experience.

We can create the visualization using HTML. We could use JavaScript and C# and create a functional prototype, but that is beyond the scope of the thesis. Using HTML and JSON, we can create an interactive visualization (see Attachment H). The best way to share it is to use a common cloud code sharing database like GitHub or GitLab. We will use GitHub and upload the HTML file there (see [83]).

The interactive visualization is related to the state diagrams in Section 5.1 and Figure 34. The state diagram shows how the users can move through the pages. We excluded some connections between the ticket management and other pages in the last iteration because the diagram might be already confusing. Quick buttons perform the missing connections. Since we used the Enterprise Architect (EA) to model the system, we can save it as an XML file. Because we modelled the first four releases of the system, each has its model. So there are four files (see Attachments G), and we uploaded them on the cloud, too (see [83]). This way, if we have EA on our computer, we can browse the model there.

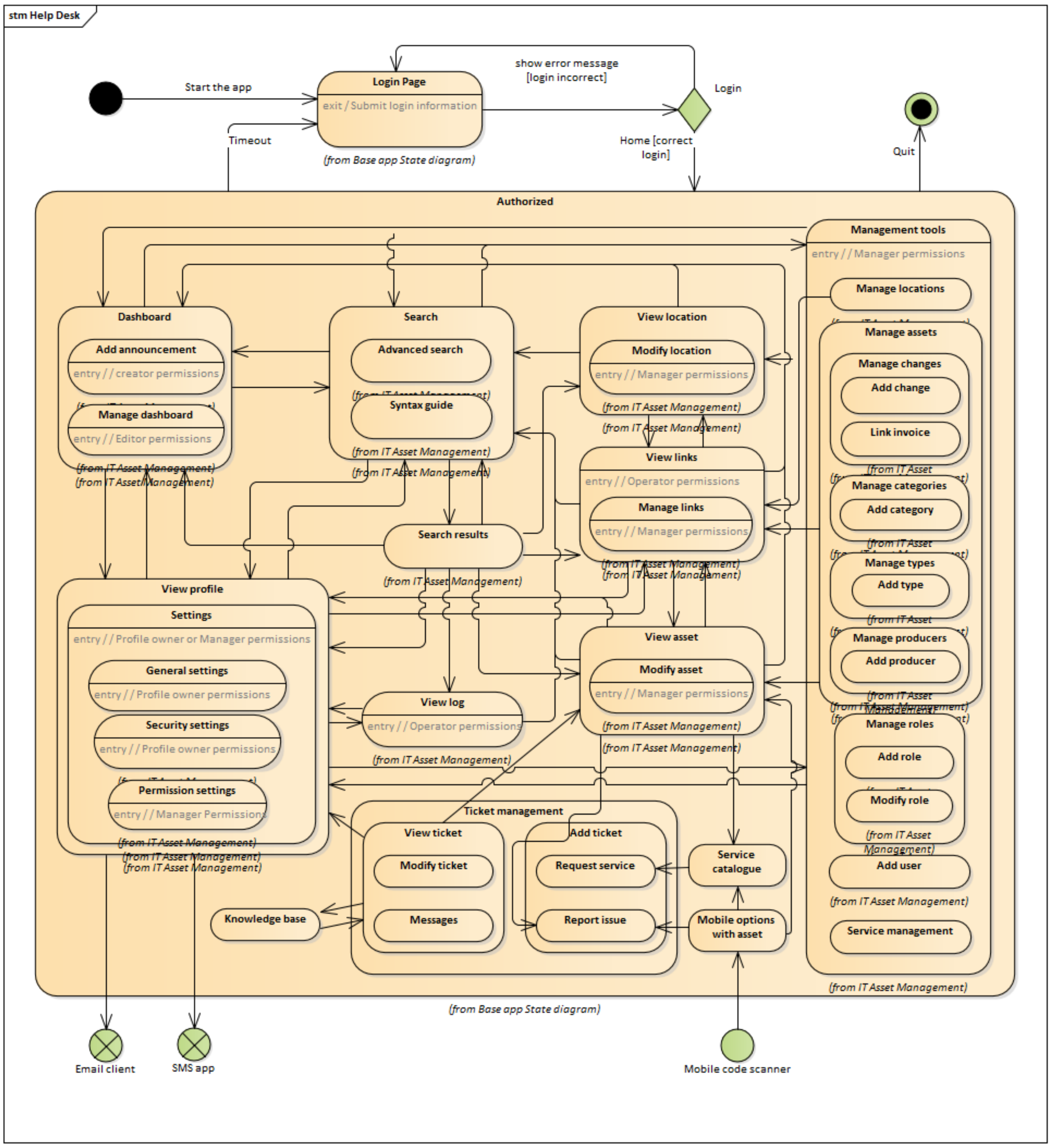

Figure 34: State diagram of help desk release of ANS CR single help desk

6. Next steps

In the previous chapter, we created a model of the ANS CR single help desk application. We also presented a proposal of the time plan, and we proposed a work breakdown structure (WBS) and risks for the project. The project manager may use all those proposals and start working on the project right away.

On the other hand, the first step should be to use the visualization and update the requirements. The IT analyst should look at the model, and if there is any information missing, they should fill them in. This step highly depends on the organizational culture, personal preferences, and experiences. For some organizations, the model we created might be perfect, but the model might be missing something for others.

The project manager must create an operational and specific time plan. They can use our proposal as a template, but it is still missing the specific time. The project length and WBS depend on the development team. Some teams might be pleased with it and just add times to the time plan. The time in the time plan highly depends on the development team. Some teams might prefer different time plans.

Most importantly, the team should consist of experienced developers because *it takes an IT village to raise a service desk* [85]. We expect the development team to be using the agile project management method, which makes risk management difficult. On the other hand, if they closely cooperate with the customers and focus on one project at a time, they should be just fine.

The project manager must have good release management. If they use the WBS we suggested, they must continuously teach the users how to work with it and listen to the feedback.

When the development team finishes the system we modelled, they might move to a new project or improve the system further. In the WBS, we already suggested few ways how to improve it.

The **availability management** is the first possible improvement of the system. We described its value in Section 5.1.5. The development team might use the entry system, which logs the entries and exits if the user used their entry card. The available application can then show the available personnel to the managers while they are assigning tickets. The system can also use this data to assign the ticket to the available manager correctly.

In the **following improvements** WBS, we suggested several functionalities (see Section 5.1.6). The improvements are focused primarily on improving the customer experience for the managers and technicians. The inspiration for the availability improvement and following improvements came from help desks and service desks already available and successful solutions on the market. There will be other improvement ideas, which will come from the users. We can already suggest few functionalities, which the users might appraise.

The system might allow the users to **add images to the asset changes**. That goes hand in hand with the state of the assets. The database can track the state of an asset, but it is always better to prove it with an image. It is also a reasonable precaution against thefts and or losses. If there is a hard proof of controller exchange, the only way how to lose the asset is for the user to forget it somewhere.

The application can also allow the users to modify the layout of the dashboard. The **settings for the dashboard** could also allow the users to prioritize some tags over others. That means that the user would see announcements with specific tags or authors in a particular part of the page.

There might be multiple users who created the same ticket. If the operators could **group the tickets**, it would make their life easier, and the knowledge base would not have to store so much data. It might also be interesting to introduce types to the issues. That means that the author of the issue might decide which type the issue is. Example groups might be hardware, software, and others. This functionality would demand just one new entity in the database, a new page in the management tools, and a new choice line in the issue reporting. The value of this functionality is in the automatic assigning to the managers. Meaning that the application can better automatically assign the tickets if it knows who specializes in what.

All the functionalities, which we talked about in this section are optional. For some organizations, it would make the system too robust, but for others, it is necessary. We have skipped their modelling just because of this reason.

7. Conclusion

In this work, we aimed to create the specification, project management, and model for the help desk system with asset management for the Air Navigation Services of the Czech Republic. In the first part of the thesis, we investigated the theoretical base. We defined the service desk, help desk and IT asset management terms. The descriptions of possible project management methodologies and project management activities are also there. Finally, there is a description of the analysis and modelling tools. We used those tools in subsequent parts.

The specification includes the requirements for the system and a description of the desired product. We defined the requirements using the current situation analysis, market research, and interviews. Since the development will be internal, the requirements will probably change. The development team must listen to the feedback and adjust the requirements as well as the model.

We suggested the work breakdown structure (WBS), time plan, and risk register with possible mitigations in the next section. The recommended project management methodology is agile though we modelled the system with fixed requirements. Our WBS divided the system into releases. The first one is the base app, which allows the users to log in, modify their profile, and add new users. The second release adds a dashboard, and it is called the service desk. The third release is called IT asset management, and it introduces the locations and assets to the system. The users can control assets and have offices. At the same time, the managers can add new assets, record changes, and link them with invoices from accounting applications. Finally, the help desk release allows the users to report issues and request services. The managers can manage the tickets.

The model consists of class diagrams, use case diagrams, activity diagrams, state machine diagrams, and entity-relationship diagrams (ERD). The diagrams are in attachments, except for the use case diagrams and the state machine diagrams. Each release until the help desk system has a model.

We created the unified modelling language model using the Enterprise architect. That means that we exported it as XML and saved it on the cloud. We did the same with the database, saving it as an SQL file onto the cloud. Additionally, we created an interactive visualization, which might help with updating the requirements. We saved the visualization as an HTML file on the cloud too. We also included those files as Attachments.

Finally, we proposed the next steps after the thesis. Some of the steps are necessary for the project, but some are the following improvements. We described the improvements, but there is no further analysis or model for them.

In conclusion, the model we created is universal for any customer. If an organization wants an independent help desk with IT asset management and service desk, they can use this thesis. Since we created a project management proposal, the organization must adjust all the parts to fit their requirements. On the other hand, it will save them time and resources. It is possible to deepen the model by adding actions which the Enterprise Architect can convert into a code.

References

[1] Miller, S. P. (2019) 'Information Technology Infrastructure Library (ITIL)', *Salem Press Encyclopedia*. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=ers&AN=89550589&site=eds-live&scope=site (Accessed: the 25th of February 2021).

[2] Dryden, S. (2019) 'Effective Implementation of ITIL/ISO 20000 Problem Management'. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=edsbas&AN=edsbas.98E2ED23&site=eds-live&scope=site (Accessed: the 25th of February 2021).

[3] Media, C. (2016) *ITIL for Beginners: The Complete Beginner's Guide to ITIL, Second Edition*. Second Edition. ClydeBank Media LLC. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1095162&site=eds-live&scope=site (Accessed: the 26th of February 2021).

[4] Axelos (2019) *ITIL Foundation: ITIL 4 Edition*. The Stationery Office. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1115788&site=eds-live&scope=site (Accessed: the 2nd of March 2021).

[5] Holt, J. (2004) *UML for systems engineering : watching the wheels, second edition*. 2nd ed. Institution of Electrical Engineers (IEE professional applications of computing series: 4). Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1068186&site=eds-live&scope=site (Accessed: the 15th of March 2021).

[6] Rumpe, B. (2017) Agile modeling with UML : code generation, testing, refactoring. Springer. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.897972&site=eds-live&scope=site (Accessed: the 15th of March 2021).

[7] Shoemaker, M. L. (2004) *UML applied : a .NET perspective*. Apress. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1067415&site=eds-live&scope=site (Accessed: the 15th of March 2021).

[8] Chonoles, M. J. and Schardt, J. A. (2003) *UML 2 for dummies*. Wiley (--For dummies). Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1066862&site=eds-live&scope=site (Accessed: 25 July 2021).

[9] Ó Conchúir, D. (2010) *Overview of the PMBOK: registered: guide : short cuts for PMP certification*. Springer. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1072299&site=eds-live&scope=site (Accessed: the 18th of March 2021).

[10] Agutter, C. (2020) *ITIL 4 Essentials: Your Essential Guide For the ITIL 4 Foundation Exam and Beyond, Second Edition*. Second Edition. IT Governance. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1115781&site=eds-live&scope=site (Accessed: the 18th of March 2021).

[11] AXELOS and Great Britain Cabinet Office (2012) 'Essential ITIL: processes and functions'. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=edsbas&AN=edsbas.5B290B4&site=eds-live&scope=site (Accessed: the 6th of April 2021).

[12] AXELOS and Great Britain Cabinet Office (2012) 'Introduction to the ITIL service lifecycle'. Available at: https://search-ebscohost-

com.e.bibl.liu.se/login.aspx?direct=true&db=edsbas&AN=edsbas.4F8CD360&site=eds-live&scope=site (Accessed: the 9th of April 2021).

[13] Office, T. S. and Office, T. S. (2016) *ITIL Practitioner Guidance.* The Stationery Office. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1131101&site=eds-live&scope=site (Accessed: the 11th of April 2021).

[14] Herzog, M. V. and Stelling, D. (2021) 'Flight operation officers: From job analysis to selection procedures', *Aviation Psychology and Applied Human Factors*, 11(1), pp. 48–53. doi: 10.1027/2192-0923/a000202.

[15] Jan Štůla (2019) 'Návrh softvérového rozhraní pro OCC simulátor ; Design of a Software Interface for OCC Simulator'. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=edsbas&AN=edsbas.490BBD93&site=eds-live&scope=site (Accessed: 24 June 2021).

[16] Ferrara, R. J. (2020) 'Air traffic control', *Salem Press Encyclopedia of Science*. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=ers&AN=89250350&site=eds-live&scope=site (Accessed: the 24th of June 2021).

[17] '*Protocol on the accession of the European Community to the Eurocontrol International Convention relating to Cooperation for the Safety of Air Navigation of the 13th of December 1960, as variously amended and as consolidated by the Protocol of the 27th of June 1997*' (2004) Official Journal L 304/210 (Accessed: the 26th of June 2021).

[18] Air Navigation Services of the Czech Republic (2020) '*Annual report 2019*'. Available at: http://www.rlp.cz/en/company/performance/Pages/reports.aspx (Accessed: the 28th of June 2021)

[19] EUROCONTROL (2020) '*Daily Traffic Variation – States*'. Available at: https://www.eurocontrol.int/Economics/DailyTrafficVariation-States.html (Accessed the 28th of June 2021)

[20] Atlassian 2021, *Service desk vs help desk vs ITSM: What's the difference?*, Atlassian, viewed the 1st of June 2021, https://www.atlassian.com/itsm/service-request-management/help-desk-vs-service-desk-vs-itsm

[21] Alstanet, s.r.o., 2021, *Produkty / Facility Management Software – AFM*, Alstanet, s.r.o., viewed 1 June 2021, http://www.alstanet.cz/Clanek/Produkty/Facility-Management-Software-AFM/AFM-Alstanet-Facility-Management-Software/3031.aspx

[22] BMC Software, Inc., 2021, *BMC Helix ITSM is the next generation of Remedy*, BMC Software, Inc., viewed the 1st of June 2021, https://www.bmc.com/it-solutions/remedy-itsm.html

[23] Oliver Rist, 2021, *The Best Help Desk Software for 2021*, PCMag, viewed the 4th of June 2021, https://www.pcmag.com/picks/the-best-help-desk-software

[24] Capterra, 2021, *Help Desk Software*, Capterra, Inc., viewed the 4th of June 2021, https://www.capterra.com/help-desk-software/

[25] Christopher Robinson, 2021, *20 Best Help Desk Software Solutions of 2021*, Finances Online, viewed the 4th of June 2021, https://financesonline.com/top-20-help-desk-software-solutions/

[26] LinkedIn, 2021, *Jira in Czechia*, LinkedIn Corporation, viewed the 4th of June 2021, https://www.linkedin.com/jobs/search/?keywords=JIRA

[27] Bulb Digital, 2020, *Help Desk App – Free with Microsoft 365 Subscription*, YouTube, viewed the 4th of July 2021, https://www.youtube.com/watch?v=q03MaN2bNXc

[28] Christopher Robinson, 2021, *15 Best Service Desk Software Solutions*, Finances Online, viewed the 4th of June 2021, https://financesonline.com/top-15-service-desk-software-solutions/

[29] Olya Kurinna, 2019, *Top 16 Service Desk Software Solutions for Your IT Company Ranked*, Help Desk Migration, viewed the 4th of July 2021, https://help-desk-migration.com/top-service-desk-software-solutions-for-your-it-company-ranked/

[30] Wienclaw, R. A. (2021) 'Project Management', *Salem Press Encyclopedia*. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=ers&AN=89163935&site=eds-live&scope=site (Accessed: the 5th of July 2021).

[31] 'Work Breakdown Structure' (2021) *Salem Press Encyclopedia*. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=ers&AN=148527209&site=eds-live&scope=site (Accessed: the 5th of July 2021).

[32] Project Management Institute (2019) *Practice Standard for Work Breakdown Structures, Third Edition*. Third Edition. Project Management Institute. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1054484&site=eds-live&scope=site (Accessed: the 5th of July 2021).

[33] Project Management Institute (2017) *Agile Practice Guide*. Project Management Institute. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1078044&site=eds-live&scope=site (Accessed: the 5th of July 2021).

[34] Maha Al-Kasasbeh, Osama Abudayyeh and Hexu Liu (2020) 'A unified work breakdown structure-based framework for building asset management', *Journal of Facilities Management*, 18(4), pp. 437–450. doi: 10.1108/JFM-06-2020-0035.

[35] Codington-Lacerte, C. (2020) 'Agile software development', *Salem Press Encyclopedia*. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=ers&AN=90558239&site=eds-live&scope=site (Accessed: the 5th of July 2021).

[36] Cooke, J. L. (2016) *Agile : an executive guide : real results from it budgets*. Second edition. IT Governance Publishing. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1107597&site=eds-live&scope=site (Accessed: the 5th of July 2021).

[37] Thesing, T., Feldmann, C. and Burchardt, M. (2021) 'Agile versus Waterfall Project Management: Decision Model for Selecting the Appropriate Approach to a Project', *Procedia Computer Science*, 181, pp. 746–756. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=edo&AN=148883827&site=eds-live&scope=site (Accessed: the 6th of July 2021).

[38] Axelos (2018) *PRINCE2 Agile*. The Stationery Office. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1115810&site=eds-live&scope=site (Accessed: the 6th of July 2021).

[39] Harmon, A. (2020) 'Gantt chart', *Salem Press Encyclopedia*. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=ers&AN=119214069&site=eds-live&scope=site (Accessed: the 6th of July 2021).

[40] Kumar, P. P. (2005) 'Effective Use of Gantt Chart for Managing Large Scale Projects', *Cost Engineering*, 47(7), pp. 14–21. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=buh&AN=17721066&site=eds-live&scope=site (Accessed: the 6th of July 2021).

[41] Herz, T. P. et al. (2013) 'Toward a model of effective monitoring of IT application development and maintenance suppliers in multisourced environments', International

Journal of Accounting Information Systems, 14(3), pp. 235–253. doi: 10.1016/j.accinf.2012.12.003.

[42] Harmon, A. (2020) 'SWOT analysis', *Salem Press Encyclopedia*. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=ers&AN=100259317&site=eds-live&scope=site (Accessed: the 10th of July 2021).

[43] Dziak, M. (2020) 'Brainstorming', *Salem Press Encyclopedia*. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=ers&AN=100259213&site=eds-live&scope=site (Accessed: the 11th of July 2021).

[44] Ashbaugh, D. A. (2009) *Security software development : assessing and managing security risks*. CRC Press. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1070430&site=eds-live&scope=site (Accessed: the 22nd of July 2021).

[45] Podari, Z. et al. (2020) 'Systematic Literature Review on Global Software Development Risks in Agile Methodology', *2020 8th International Conference on Information Technology and Multimedia (ICIMU), Information Technology and Multimedia (ICIMU), 2020 8th International Conference on*, pp. 231–236. doi: 10.1109/ICIMU49871.2020.9243311.

[46] Ewusi-Mensah, K. (2003) *Software development failures : anatomy of abandoned projects*. MIT Press. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1066878&site=eds-live&scope=site (Accessed: the 22nd of July 2021).

[47] Hammad, M., Inayat, I. and Zahid, M. (2019) 'Risk Management in Agile Software Development: A Survey', *2019 International Conference on Frontiers of Information Technology (FIT), Frontiers of Information Technology (FIT), 2019 International Conference on*, pp. 162–1624. doi: 10.1109/FIT47737.2019.00039.

[48] McManus, J. (2004) *Risk management in software development projects*. Elsevier/Butterworth-Heinemann. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1070537&site=eds-live&scope=site (Accessed: the 22nd of July 2021).

[49] Molina, A. D. (2016) *Ten Recommendations for Managing Organisational Integrity Risks*. The Center for The Business of Government. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1043999&site=eds-live&scope=site (Accessed: the 22nd of July 2021).

[50] Lupton, D. (2013) *Risk*. 2nd ed. Routledge (Key ideas). Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1023899&site=eds-live&scope=site (Accessed: the 22nd of July 2021).

[51] Schulmerich, M., Leporcher, Y.-M. and Eu, C.-H. (2014) *Applied asset and risk management : a guide to modern portfolio management and behavior-driven markets*. Springer (Management for Professionals). Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.852795&site=eds-live&scope=site (Accessed: the 22nd of July 2021).

[52] Engemann, K. J. and Henderson, D. M. (2012) *Business continuity and risk management. essentials of organisational resilience*. Rothstein Associates Inc., Publisher. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.752464&site=eds-live&scope=site (Accessed: the 22nd of July 2021).

[53] Vitaliy Kononenko, 2020, *7 MAIN TYPES OF SOFTWARE DEVELOPMENT RISKS*, Computools, viewed the 22nd of July 2021, https://computools.com/software-development-risks/

[54] Illia Kovtunov, 2021, *SOFTWARE DEVELOPMENT RISKS*, CodeIT, viewed the 22nd of July 2021, https://codeit.us/blog/software-development-risk

[55] Podeswa, H. (2010) *UML for the IT business analyst : a practical guide to requirements gathering using the Unified Modeling Language, second edition*. 2nd ed. Course Technology. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1071918&site=eds-live&scope=site (Accessed: 25 July 2021).

[56] Jordan Hirsch, 2013, *10 Steps To Successful Requirement Gathering*, Phase2 Technology, viewed on the 25th of July 2021, https://www.phase2technology.com/blog/successful-requirements-gathering

[57] Axelos (2017) *PRINCE2 Handbook*. The Stationery Office. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1131084&site=eds-live&scope=site (Accessed: 25 July 2021).

[58] Dori, D. et al. (2020) ‘System Definition, System Worldviews, and Systemness Characteristics’, *IEEE Systems Journal*, Systems Journal, IEEE, 14(2), pp. 1538–1548. doi: 10.1109/JSYST.2019.2904116.

[59] Caffrey, C. (2019) ‘Universal systems model’, *Salem Press Encyclopedia*. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=ers&AN=98402231&site=eds-live&scope=site (Accessed: 25 July 2021).

[60] Pastor, Ó. and Molina, J. C. (2007) *Model-driven architecture in practice. a software production environment based on conceptual modeling*. Springer. Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.624428&site=eds-live&scope=site (Accessed: 25 July 2021).

[61] Gottardi, T. and Vaccare Braga, R. T. (2019) ‘Evaluating the Ability of Developers to Use Metamodels in Model-Oriented Development’*, 2019 IEEE/ACM 11th International Workshop on Modelling in Software Engineering (MiSE)*, Modelling in Software Engineering (MiSE), 2019 IEEE/ACM 11th International Workshop on, pp. 27–34. doi: 10.1109/MiSE.2019.00012.

[62] Zimmerová, L. and Beránek, L., 2021, *Tender Documentation – Podniková platforma řízení zdrojů Etapa Finance*, NEN, viewed on 27th of July 2021, https://nen.nipez.cz/RegistrZadavatelu/RegistrZakladniUdajeZadavateleM-185358799/RegistrZakladniUdaje-185358799/MultiprofilZakladniUdajeOZadavateliM-191312505=206522948/SeznamZahajenychZadavacichPostupu-191312505/ZakladniInformaceOZadavacimPostupuM-1074585889-191312504/ZadavaciDokumentace-1074585889-191312504/

[63] *The Technology Intelligence Platform*, Snow software, viewed on 27th of July 2021, https://www.snowsoftware.com/

[64] 2021, *Functional and Nonfunctional Requirements: Specification and Types*, Altexsoft, viewed on 27th of July 2021, https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/

[65] Czechowski, A. and olprod, 2019, *Co je Configuration Manager?*, Microsoft, viewed on 27th of July 2021, https://docs.microsoft.com/cs-cz/mem/configmgr/core/understand/introduction

[66] 2021, *Best Service Desk Software*, g2, viewed on 27th of July 2021, https://www.g2.com/categories/service-desk
[67] 2021, *Top 10 Best IT Asset Management Software In 2021 (Pricing And Reviews)*, Software Testing Help, viewed on 27th of July 2021, https://www.softwaretestinghelp.com/it-asset-management-software/
[68] Richard Best, 2021, *Best Asset Management Software*, Investopedia, viewed on 27th of July 2021, https://www.investopedia.com/best-asset-management-software-5090064
[69] 2021, *IT Asset Management Software*, Capterra, viewed on 27th of July 2021, https://www.capterra.com/it-asset-management-software/
[70] Lanette Creamer, 2019, *The Best IT Asset Management Software*, PCMAG, viewed on 27th of July 2021, https://www.pcmag.com/picks/the-best-it-asset-management-software
[71] 2021, *What is Diagrams.net*, JGraph Ltd., viewed on 28th of July 2021, https://www.diagrams.net/
[72] 2021, *Where seeing becomes doing.*, Lucid Software Inc., viewed on 28th of July 2021, https://www.lucidchart.com/pages/
[73] 2021, *StarUML*, MKLabs Co.,Ltd., viewed on 28th of July 2021, https://staruml.io/
[74] 2021, *Enterprise Architect*, Sparx Systems Pty Ltd., viewed on 28th of July 2021, https://sparxsystems.com/products/ea/index.html
[75] Lithmee, 2019, *What is the Difference Between Class Diagram and Entity Relationship Diagram*, viewed on 28th of July 2021, https://pediaa.com/what-is-the-difference-between-class-diagram-and-entity-relationship-diagram/
[76] Indika, 2011, *Difference Between ER Diagram and Class Diagram*, Difference Between, viewed on 28th of July 2021, https://www.differencebetween.com/difference-between-er-diagram-and-vs-class-diagram/
[77] Bagui, S. and Earp, R. (2003) *Database design using entity-relationship diagrams. Auerbach (Foundations of database design: 1).* Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.1066868&site=eds-live&scope=site (Accessed: 28 July 2021).
[78] 2021, *Integromat Enterprise Cloud Service*, Integromat by Celonis, viewed on 31st of July 2021, https://www.integromat.com/en/enterprise
[79] 2021, *Connect your cloud apps. Automate work.*, Automate.io, viewed on 31st of July 2021, https://automate.io/
[80] Drupal, 2021, *Concept: Users, Roles, and Permissions*, Dries Buytaert, viewed on 31st of July 2021, https://www.drupal.org/docs/user_guide/en/user-concept.html
[81] 2021, *Scryfall Search Reference*, Scryfall, LLC., viewed on 31st of July 2021, https://scryfall.com/docs/syntax
[82] Ferraiolo, D., Chandramouli, R. and Kuhn, D. R. (2007) *Role-based access control*. 2nd ed. Artech House (Artech House information security and privacy series). Available at: https://search-ebscohost-com.e.bibl.liu.se/login.aspx?direct=true&db=cat00115a&AN=lkp.538380&site=eds-live&scope=site (Accessed: 31 July 2021).
[83] Jan Štůla, 2021, *Single-Help-Desk*, GitHub, viewed on 2nd of August 2021, https://github.com/StulaJan/Single-Help-Desk
[84] Anton Nazarov, 2019, *Why another database tool?*, Apricot, viewed on 2nd of August 2021, https://www.apricotdb.co.za/joomla/index.php
[85] Kirsten Petersen (2019) 'It Takes an IT Village to Raise a Service Desk', pp. 148–151. doi: 10.1145/3347709.3347832.

List of figures

List of tables

List of attachments