POLITECNICO MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

Hallbridger: Real-Time Acoustic Control in Multipurpose Halls

Master of Science thesis in Music and Acoustic Engineering

Author: **Jacopo Fantin**

Student ID: 103078
Advisor: Prof. Maria Cairoli
Co-advisors: Prof. Marcello Maria Bersani, Ing. Laura Tiburzi
Academic Year: 2024-25

Abstract

As technology develops, the pipe dream of unifying different buildings meant for events ranging from conferences to classical music concerts, which present very diverse acoustic requirements, in a single, multipurpose hall becomes more and more achievable, making this kind of architecture more and more common. With the help of Internet of Things (IoT) sensors and connectivity, and taking advantage of Digital Twin (DT) technologies and Building Information Modeling (BIM) processes, a context where a human operator can remotely view, modify, and perform acoustic simulations in a Virtual Reality (VT) version of a multifunctional space is not so unrealistic anymore. The present work constitutes a first step into this scenario by developing a software interface able to bridge a theater's facility management system with a 3D model of its hall, letting the digital model update, based on real-time data about the current configuration of the hall's movable elements, and letting the user visualize acoustic information (specifically, the reverberation time) accordingly. The thesis focuses on the practical application of this interface at the Roberto de Silva Theater in Rho, Milan, but is intended to be adjustable for different stage control software, and to be extended to the complex system previously described.

Keywords: multipurpose halls, multifunctional theater, internet of things, iot, digital twin, building information modeling, bim, virtual reality, vr, real-time acoustics, software interface, 3d model, ifc file, reverberation time, stage control software, facility management system, acoustic simulations, computer science

Abstract in lingua italiana

Con lo sviluppo della tecnologia, la chimera di unificare in un'unica sala polifunzionale diversi edifici destinati a eventi che spaziano da conferenze a concerti di musica classica, che presentano requisiti acustici molto diversi tra loro, diventa sempre più realizzabile, rendendo questo tipo di architettura sempre più comune. Con l'aiuto di sensori e connettività Internet of Things (IoT) e sfruttando le tecnologie di gemelli digitali (DT) e processi di Building Information Modeling (BIM), un contesto in cui un operatore umano può visualizzare, modificare ed effettuare da remoto simulazioni acustiche in realtà virtuale di uno spazio multifunzionale non è più così irrealistico. Il presente lavoro costituisce un primo passo in questo scenario sviluppando un'interfaccia software in grado di collegare la macchina scenica di un teatro con un modello 3D della sua sala, consentendo al modello digitale di aggiornarsi in base ai dati in tempo reale sulla configurazione corrente degli elementi mobili della sala e permettendo all'utente di visualizzare di conseguenza le informazioni acustiche (in particolare, il tempo di riverbero). La tesi si concentra sull'applicazione pratica di questa interfaccia al Teatro Roberto de Silva di Rho, Milano, ma è pensata per essere adattabile a diversi software di controllo scenico e per essere estesa al complesso sistema descritto in precedenza.

Parole chiave: sala polivalente, teatro multifunzionale, internet of things, iot, gemello digitale, building information modeling, bim, realtà virtuale, acustica in tempo reale, interfaccia software, ingegneria informatica, modello 3d, file ifc, tempo di riverbero, macchina scenica, simulazioni acustiche

Contents

Introduction

As technology develops and human standards get better by raising the level of quality life, sustainability and social environment, the need of buildings serving more than one purpose increases. Multifunctional halls are becoming a new way to conciliate social and cultural life of citizens while following a more sustainable approach, both in terms of energy consumption and land build upon [1, 2, 7]. However, effectively and efficiently uniting more functions in one single building has been regarded more as wishful thinking than as a concrete possibility in room acoustics until the beginning of this century [3]: the required technologies and maintenance costs have been considered too consistent to constitute a feasible solution for new buildings devoted for shows, concerts and events, and the resulting acoustics too poor to satisfy high standards and to make the elevated costs worth the work. Materials availability and architecture techniques in use nowadays have finally let this kind of multifunctional structures be a tangible option for theaters, music halls and auditoria, other than a more and more promising one for the future, to the point when they are likely to become the standard for cultural and social event venues in general.

This current scenario generated a new need for analyzing and organizing the available equipment dedicated to a venue's specific purpose, commonly referred to as facility management (FM), in a smart fashion. When applied to the building category this thesis focuses on, FM involves architectural and functional elements capable of changing the room acoustics of the inner space, to accommodate the different functions it was envisioned for. In fact, it can be easily understood that a congress hall requires a very limited quantity of echoes and reflections between walls for the human voice to be as intelligible as possible, whereas a classical music concert is better enjoyed in a more reflective environment. To modify the quantity of reflected sound waves and the acoustic perception in general, the most immediate approach is to employ movable structures that have an impact on how sound is dissipated, which could be achieved by exposing absorbing rather than reflecting materials or orienting surfaces in different directions. Worth mentioning, variable acoustics allows for rehearsals in conditions similar to the ones by which the artistic performance is going to take place. In particular, seats occupied by the audience, which is present in the latter and absent in the former, usually have a considerable effect on the amount of acoustic energy absorbed and not reflected back to the space boundaries. The setup of such multipurpose spaces can benefit from facility management practices to optimize resources and ease the operators' job to match the staging process with the activity to be carried out next in the hall.

It is not by chance that FM represents one crucial element of Building Information Modeling (BIM), a modern way of conceiving structure and infrastructure design, construction, utilization and maintenance, and particularly for these last two. The BIM approach allows a system to be more efficiently kept and exploited, crossing data that belong to different aspects of the project, but that influence each other and are connected one another to have an impact on its management. Automation and smart strategies in the hall preparation are just two of the possibilities that a BIM system enables when our multifunctional context is part of it.

Meanwhile, Internet of Things technologies keep on spreading in the industry and research world as well as in the private citizens' sphere. With the possibility of communication between buildings' control system software and sensors, connected devices or pieces of equipment, IoT applications are a potentially huge aid to facility management, especially in rooms with moving elements and variable acoustics. Combining real-time updates from IoT devices about the physical environment (temperature, air density, background noise, etc.) with a digital representation of the hall, and making this combination part of a BIM environment, we are able to complete the live description of the venue in question given by its FM software. Most of all, this way we can obtain all the information needed to evaluate the hall's acoustics: in fact, its digital representation, kept in step with the real situation, can be devised to be oriented to acoustic simulations, so to test virtually how the room reacts to a specific sound source inside of it depending on how we position its equipment, without having to do it on site, with real sound sources and trying out different listener points. This can speed up the entire room preparation procedure by a great deal [3], leaving the involved professionals (orchestra conductors, theater directors, musicians, actors, technicians, etc.) more time and energy for case-specific fine-tuning of the acoustics. The integration between all the members we mentioned in a BIM system could lead to scenarios such as the one depicted in figure fig. 1 on page xii: IoT devices provide their output, labeled with the wide term "physical context", to the facility management system, which "reads" the current hall configuration, being connected to the moving elements it controls; all available data are forwarded to a software interface, potentially located remotely with respect to the real hall, that allows for linking the real hall to the virtual one and that in turn injects the data into the hall's 3D model, modifying it (blue arrow path in the figure); a theater operator can view the virtual hall fed with the 3D model, move around architectural and stage structures and perform evaluations on the hall acoustics[1]. The gray arrows in this portion of the scheme denote a "control flow" piloting the user's possible actions. Finally, they can confirm the virtual hall configuration the moment they are content with the outcome, sending the chosen layout to the theater's facility management system to actually let the hall set up correspondingly, through the inverse data flow (yellow arrow path).

Despite the great usefulness of this integration, so far it has hardly been applied to existing systems, given the obstacles in making on-site devices communicate with BIM models, especially in a dynamic way: there exists no standardized protocol to implement this kind of connection, due to the heterogeneity of the pieces of equipment that vary from case to case [3]. Customized solutions ("Ad-hoc software interface" in 1) should be thought of and developed as general and applicable to different contexts as possible, in an attempt to ease the adoption of Industry 4.0 in acoustics by new and existing buildings. By this, we mean to exploit the new possibilities that Internet of Things and smart connected devices give us to automate the theater setup and to aid workers and operators in evaluating the best configuration for the hall's next employment, thanks to real-time data and predictive algorithms.

[1]This step is left in generic terms for the moment: it could take place through some external acoustic simulation software, or one inside a digital twin system (as assumed in 2.2), or just by matching the current hall configuration with one for which we have some recordings and acoustic information at disposal, performing data interpolation in case it doesn't match any of them.

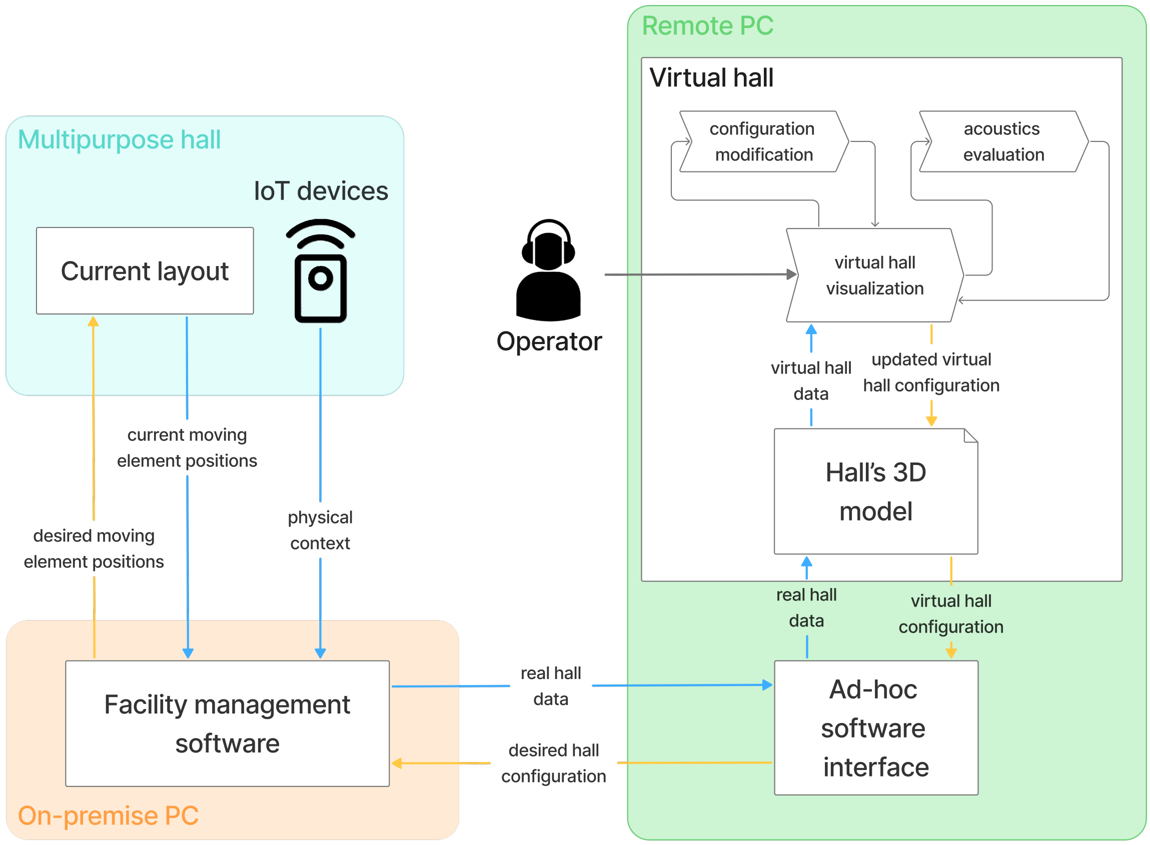


Figure 1: Example scheme of integration between real and virtual hall through an appropriate software interface.

The context of modern buildings that can embrace Building Acoustics 4.0 is where the Roberto de Silva theater in Rho, Milan, was conceived. This venue is the starting point of the project idea introduced in [3], which we just have quickly generalized. The aim of the hereby presented thesis is to create the aforementioned ad-hoc piece of software, called Hallbridger, a first building block of the whole project idea that is able to put in a bidirectional communication this theater's facility management software, CURIO ©, by the stage equipment company *Decima 1948*, based in Padua, Italy, with a digital tridimensional model of the theater's hall. This first step alone won't implement a system that can be called a BIM system, but it is conceived so that a future version of it will.

The work is presented as follows.
Chapter 1 introduces the Roberto de Silva theater of Rho, province of Milan, describing how it is acoustically characterized and presenting the data used by the software application.
Chapter 2 introduces BIM as a work approach and the role of the developed software in

the hypothetical physical theater-virtual theater system from the point of view of a BIM process. Chapter 3 describes and documents the interface software that is the focal point of the work.
Chapter 4 sums up the thesis content and mentions possible further developments for the software, in order to make headway in obtaining a complete tool that realizes the final scenario depicted in 1.

1 | Characterization of the Roberto de Silva theater's multipurpose hall

The Roberto de Silva theater in Rho, Milan, is a 2022 multipurpose structure providing a new gathering place for the community, among with a newbuilt square in front of it, as well as a cultural venue intended to be acoustically suitable for a number of events, even for those traditionally not carried out in a theater. We use this chapter to focus on those aspects of the theater that are related to its acoustic design, which gives the physical context where the software is intended to work.

1.1. Structure and design elements of the hall

1.1.1. Fixed architectural elements

The main hall has a parterre gallery (referred to as "gallery 0") consisting of seats close to the walls on the sides of the stalls. Galleries 1 and 2 are deployed at first and second floors all around the space, except for the wall behind the stage. Due to the house curtains, when these are pulled down and being used, the seats on the lateral walls behind the curtains cannot be utilized by the audience. The rear wall behind the stage, too, is provided with seats (referred to as "back gallery"), just above a direct access to the backstage.

Lateral walls are occupied by rotating panels, described later in section 1.1.2 on page 5, which can be considered the core of the acoustic design of the room.

The rear wall behind the audience, traditionally designed to block too strong late reflections back to the stalls and avoid echoes, is indeed made by absorbing material, but it is hidden by perforated wooden panels (5-mm-diameter holes), constituting a kind of resonator (resonant absorber with porous material behind) that generally targets low-mid-

Figure 1.1: The "Roberto de Silva" theater in Rho, Milan.

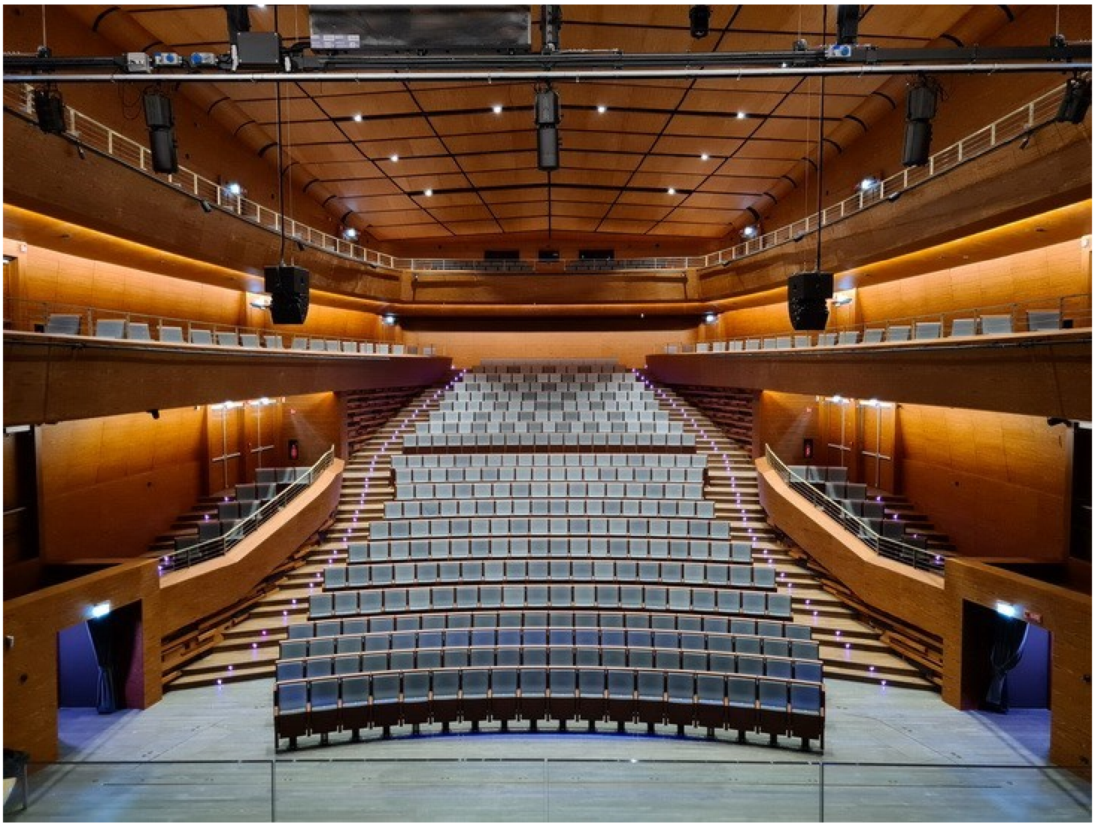


Figure 1.2: View of the stalls from the stage.

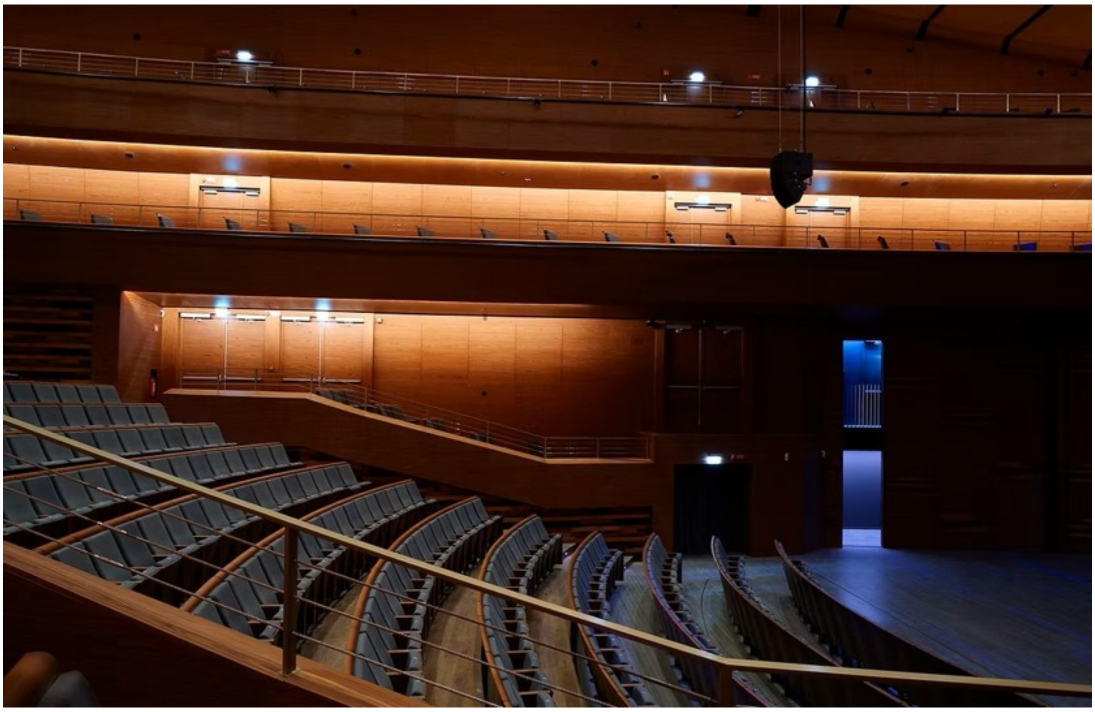


Figure 1.3: View on the hall's left wall, where the parterre is visible with its pivoting panels under the two galleries.

frequency sound waves[1]. The same design solution is adopted on portions of the lateral walls in the proximity of the stage, and around the edge of lateral walls beside entrance doors in gallery 0.

The ceiling above the stalls is provided with 20 countertop convex panels (fig. 1.4 on the following page) to ensure strong reflections from the top in all seat orders. Their density and curvature radius vary depending on the position of the panel.

Other fixed design elements that are acoustically noteworthy are the wooden coverage used for the bottom part of the lateral walls, below front and back sectors of gallery 1 and gallery 0, and for the coverage of the sliding doors below the back gallery (see section 1.1.2 on page 5): they come in various shapes and surface orientations, contributing in creating a diffuse field especially by scattering high-frequency waves upwards, to prevent them from bouncing on the floor and to keep them in the air.

It is the variable acoustics, though, that constitutes the most interesting aspect of this building.

[1]Typically between 200 and 350 Hz.

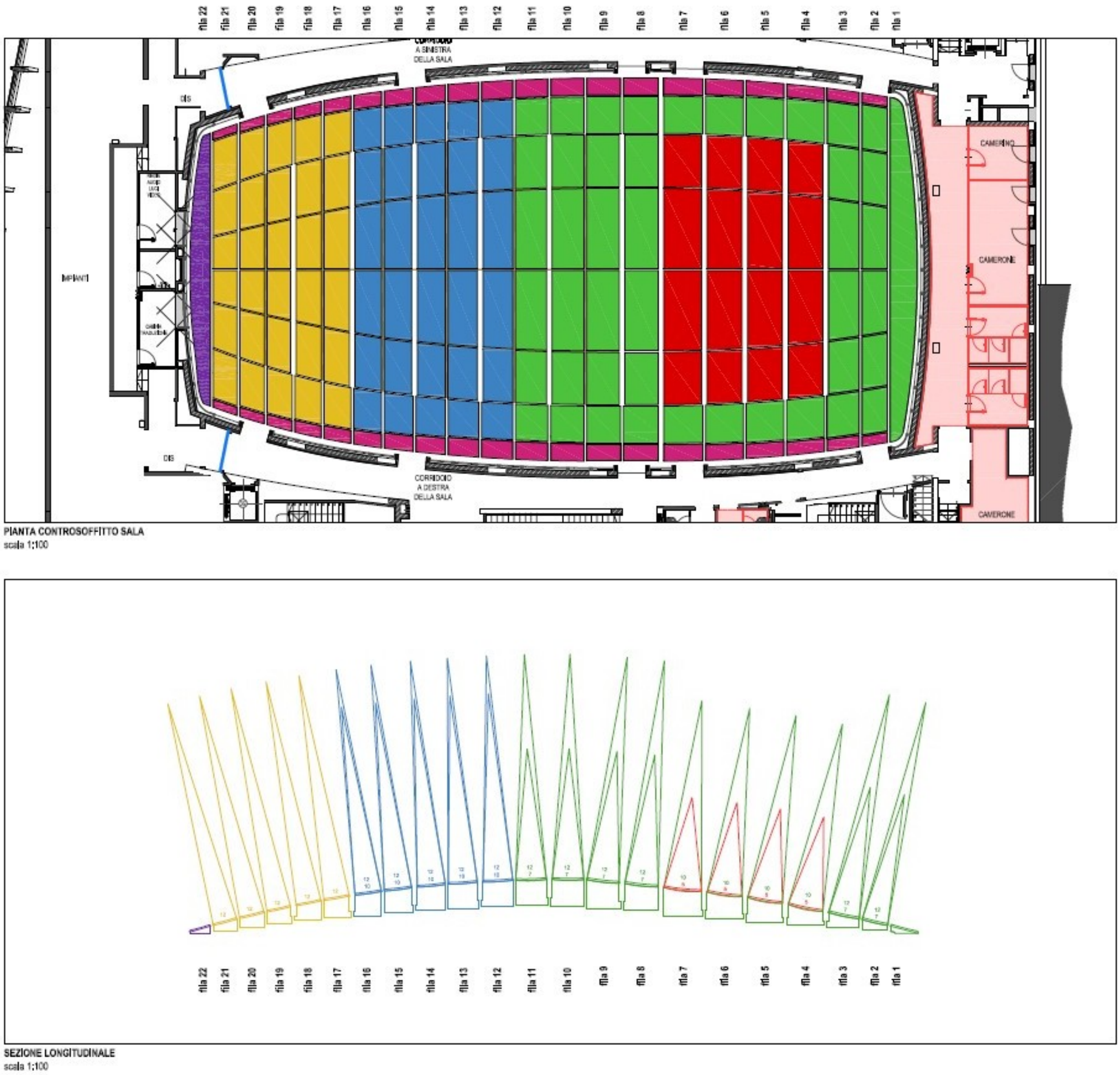


Figure 1.4: Countertop panels over the stalls. The target seats are colored the same as the corresponding panels.

1.1.2. Moving elements

The multifunctional feature of the venue is achieved through its main hall being able to change its own acoustics thanks to a set of moving and adjustable elements, which can be summarized with the following list:

- Movable architectural elements:
  - Wood ceiling panels over the stage;
  - Additional removable panels (fillers) over stage;
  - Sliding doors in the lateral walls of the stage (not automatized);
  - Sliding doors in the back wall of the stage;
  - First three seat rows in the stalls (not automatized).
- Stagecraft equipment:
  - House curtains hung on a multiline hoist;
  - Orchestra pit platform under the first three seat rows;
  - Wings and backdrops on multiline hoists;
  - Trusses on the stalls and multiline hoists on stage;
  - Stage blocks that can be lifted from the floor for scenographic purposes.
- Variable acoustic elements:
  - Pivoting panels on lateral walls.

Pivoting panels

In particular, the pivoting panels are what makes the acoustic architecture of this building peculiar. These cover the two lateral walls and are spatially organized as follows:

- 7 panel sectors on each side of the hall, whose position is represented in fig. 1.9 on page 11
- 3 or 4 panel column groupings for each sector, for a total of 26 groups for each wall and 52 in the entire hall
- one or two panel columns (or arrays) for each grouping, giving a total of 47 panel columns for both sides of the hall and 94 columns in the whole hall

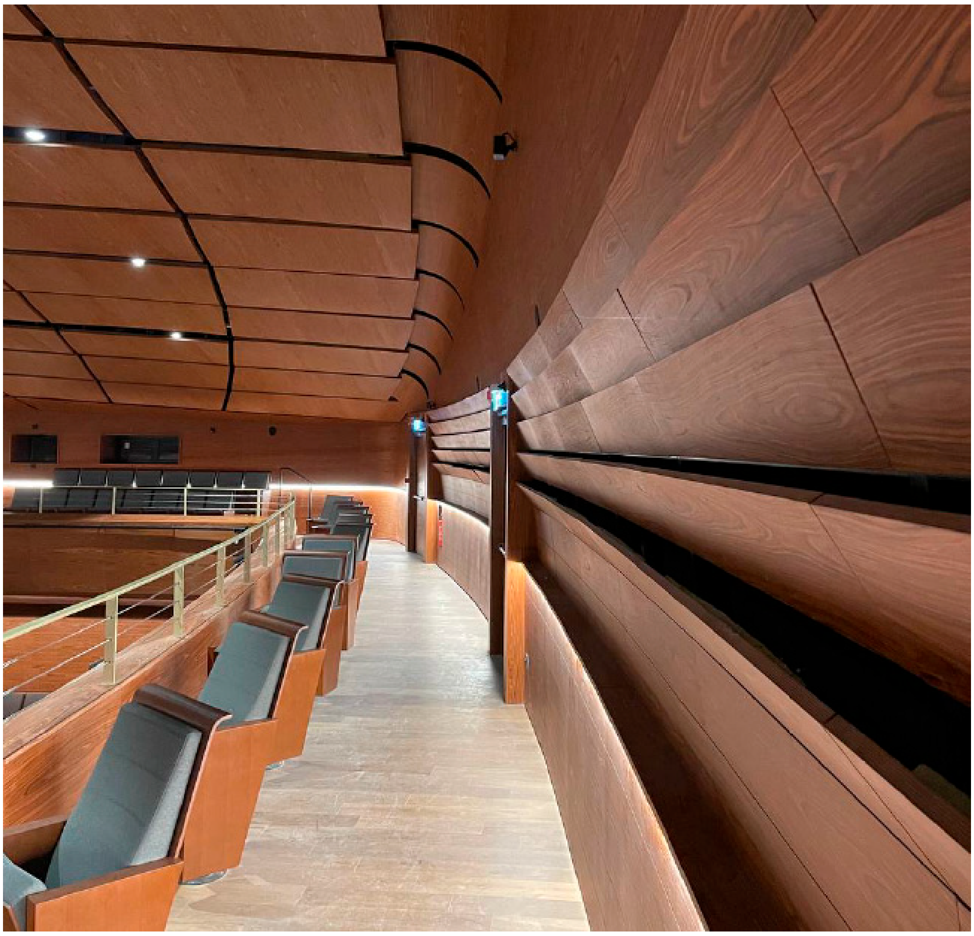


Figure 1.5: Second gallery on the left side of the hall with open pivoting panels.

- 5 panels for each panel column, depicted in fig. 1.7 on the next page, summing up to 235 for each wall and 470 in total

They are rectangular (ca. $115 \times 17 \times 2.5$ cm) wooden panels that can rotate around the bottom side, with a maximum aperture of around 45° for those in the upper part of the column and 30° for those in the lower part[2], to uncover the absorbing material placed underneath and reduce the amount of sound waves reflected by the panel's wooden surface (figure 1.6).

[2]In fact, being the lower panels of an array at head-torso height, they are able to reach a maximum aperture angle that is smaller than the one for the panels at the top. This brings us to talk about relative (to the maximum) aperture rather than in absolute terms, and to indicate it using a percentage. This feature doesn't depend on CURIO and can't be controlled from its control system: it was mechanically applied by the company that mounted the panels on the walls, upon the theater's management's request

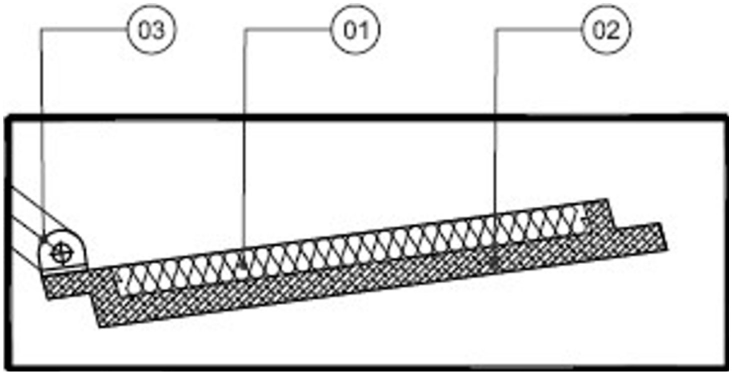


Figure 1.6: Pivoting panel scheme.

Each panel array group (so a single panel column or a couple of them, depending on the group) has a single actuator that moves the panels belonging to it, which is equivalent to saying that the array group is the panel aperture control unit: all panels belonging to the same panel array grouping have the same aperture percentage [8][3].

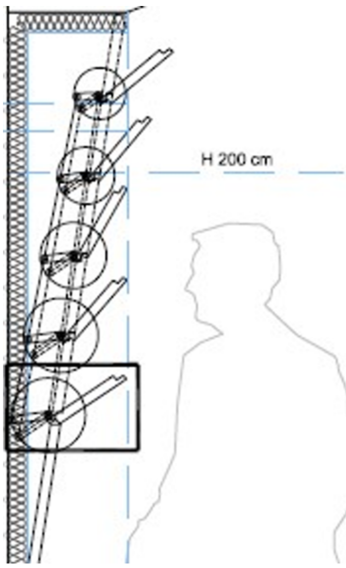


Figure 1.7: A single array of five pivoting panels.

Acoustic measurements, though, have been carried out with just some of the possible open-close combinations of the panel column groups. These measurements led to acoustic results that are briefly exposed in section 1.3 on page 10

[3]The details about this information were explained by CURIO System's manufacturer *Decima 1848*

1.2. Configurations of the multipurpose space

All of the moving elements are functional to changing the listening characteristics of the room. Although it is able to host a variety of events, shows and performances (among which cinema, drama theater and congresses [8]), three main configurations are identified [3]:

- Concert hall configuration
- Opera and ballet configuration
- Congress hall configuration

We will now go through these three main functions to describe how the various elements in the mail hall are settled to provide the corresponding layout and its acoustics [3].

1.2.1. Concert hall configuration

By this layout, the orchestra pit is closed to let the musicians play at the same level with the stalls' first rows; the total absorbing area present in the theater is minimized, so the stage curtains are removed (the lightest pieces) or lifted (the permanent ones, heavier)[4] and the pivoting panels are closed; the fly tower is closed, rotating the plywood panels over the stage to provide stronger first reflections from the ceiling; the rear sliding door gets closed to hide the backstage below the back gallery, as well as those at the two sides of the stage, enhancing first reflections from the lateral walls too. Hybrid configurations (partially electrically amplified, meaning that musicians are supported by stage monitors and their sound is captured my microphones and enhanced at low gain levels with amplifiers and loudspeakers *in situ*) are employed in case of jazz/swing concerts performed by big bands: the stage curtains are kept down or midway to obtain more absorbing material and reduce the strong reflections that loud wind instruments create[4]. Anyway, fully-electrified performances, with strong amplification, are avoided since the resulting acoustics would favor either the stalls or the back gallery: when the front loudspeakers, turned towards the stalls, are louder than the stage monitors, the back gallery receives a too dim sound compared to the front seats; when stage monitors are loud enough to optimize the acoustics for the audience in the back gallery, the one in the stalls receives too strong reflections from the back wall and perceives an undesired echo effect[4].

[4]Information provided by the Roberto de Silva theater's stage manager

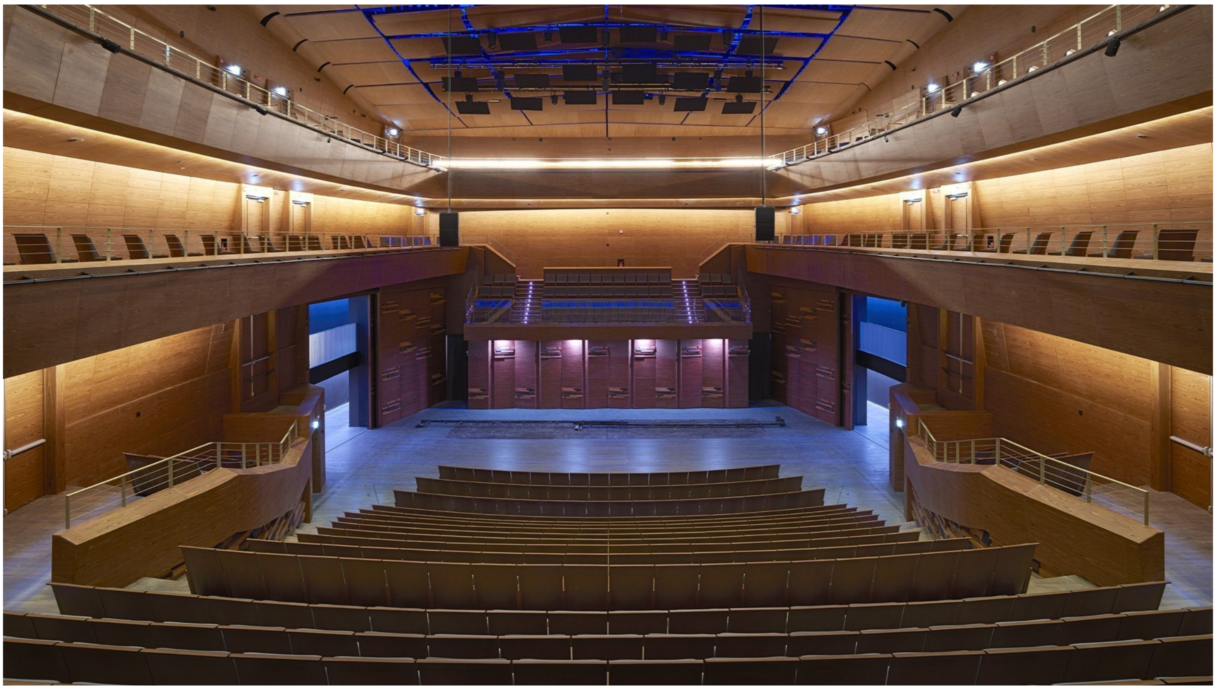


Figure 1.8: View of the stage in concert hall configuration.

1.2.2. Opera configuration

In this configuration, the first three seat rows are removed and the orchestra pit is opened for the musicians. House curtains, wings and backdrops are down and lateral and back sliding doors open for actors, singers and technicians to move freely between stage and backstage; the fly tower becomes an active acoustic space by getting opened moving the panels over stage in a vertical position.

1.2.3. Congress hall configuration

This hall layout consists in maximizing the sound-absorbing surfaces in an electro-amplified environment, to attain a speech as clear as possible from the lecturers to the audience. Hence, lateral pivoting panels are wide open, and baffles can be hung from the ceiling above the stalls to decrease reflections from the top. The orchestra pit is closed, sliding doors opened and curtains can be lowered to further boost speech clarity if reflections from the electroacoustic system are deemed to be too strong. A projection screen is hung at the back of the stage.

1.3. Acoustic measurements

To fully illustrate how the acoustic data used for the software application of this thesis were obtained, we now reference [8] to explain the details of the acoustic measurements that are of interest for our case. The procedure's aim was to obtain reverberation times in octave bands, from 125 to 4000 Hz, for various panel layouts, following the ISO 3382-1 standard [5]. Traditionally, the reverberation time T is defined as "the duration required for the space-averaged sound energy density in an enclosure to decrease by 60 dB after the source emission has stopped"[5]. However, as the standard itself makes explicit in section 3.5, T_{30}[6] can be used in place of T_{60}, and it was preferred over the latter to make the process easier.

1.3.1. Hall configuration

The most sensible open/close panel configurations to obtain relevant acoustic outcomes were regarded as those involving a whole sector, and those grouping adjacent sectors in turn. Configurations are labeled based on the sector(s) whose panels are completely open during the recordings, implying all the others are closed [8]. Configuration labels (see fig. 1.9 on the next page) follow the scheme:

- Number 0, 1 and 2 indicate parterre, first and second gallery respectively;
- Letter F indicates the back part of the hall, with arrays close to the bottom, farthest from the stage;
- Letter C stays for central sectors, with arrays in between back and front parts;
- Letter P designates the front part, with arrays above the stage.

Galleries 1 and 2 have all three back, central and front sectors, while the parterre has the central sector only. It is intended that the configurations are symmetrically applied to both lateral walls during recordings, so the corresponding panel arrays were always open/closed on both lateral walls of the theater [3]. One measurement per panel configuration was conducted.

[5]Quoted from the standard's text in Section 3.5.
[6]So, considering a decay from 5 dB to 35 dB below the initial level.

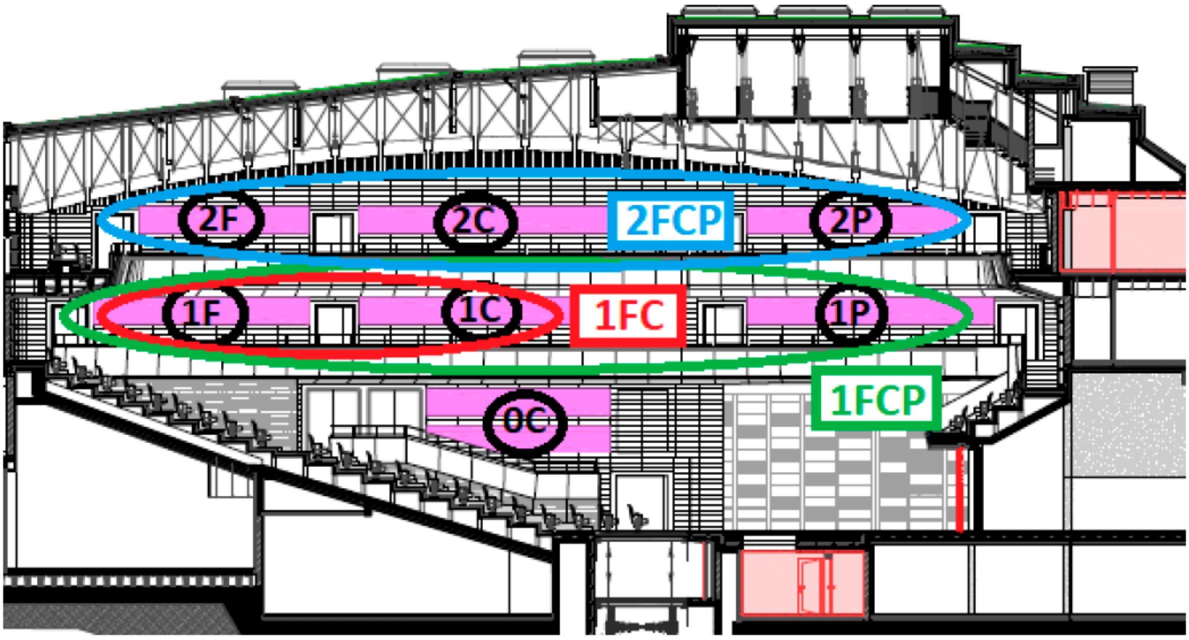


Figure 1.9: Lateral view of the theater's main hall, with pivoting panel array sectors marked in violet and labeled black, and sector groups circled and labeled in blue, red and green.

The stagecraft equipment was kept in concert hall configuration during the whole measurement phase, so the reference range for the reverberation times in octave bands is $1.5 - 1.9$ s.

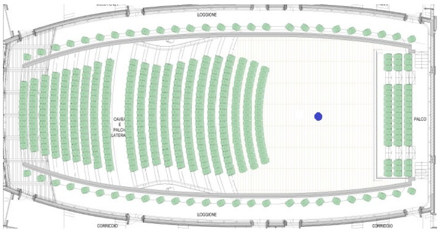
1.3.2. Measurement setup

In accordance with ISO 3382-1, the source was an omnidirectional dodecahedron speaker playing a sine sweep signal[7], placed in the middle of the stage (1.10a). Receivers were distributed in the hall as depicted in subfigures 1.10b, 1.10c and 1.10d of 1.10, so four microphones in the stalls, two in the back gallery, one in the parterre and two for each of the galleries, for a total of 11 sensors.

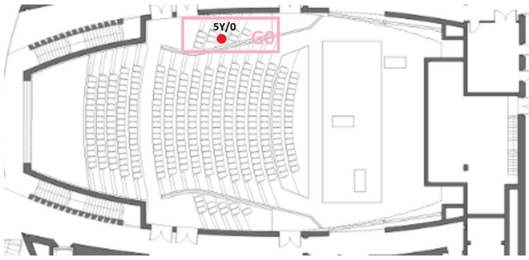
One additional recording session took place on November 4$^{\text{th}}$, 2025, again in a concert hall setting, following the same methodology that was applied for [8], just with less receiver spots and limiting them to the stalls, the parterre and gallery 1.

All measures were taken connecting the microphones with analyzer *Soundbook* by Noise-Works ©. Further details about equipment and procedures are reported in [8].

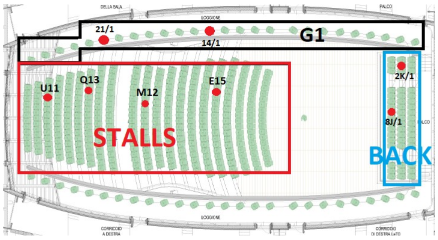
[7]Indicated as an alternative to short-transient sounds by the standard, section 3.4 [5].



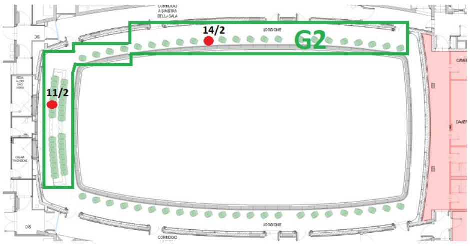
(a) Sound source position.



(b) Receiver positions (parterre/gallery 0).



(c) Receiver positions (stalls, gallery 1 and back gallery).



(d) Receiver positions (gallery 2).

Figure 1.10: Source and receiver positions.

1.3.3. Measurement results

The collected data were then processed[8] to extract various reverberation times, relative to predefined receiver groups, by averaging their signals correspondingly. The most interesting result for our application is the **global reverberation time**, that considers all the receivers at once, resulting in table 1.1 on the facing page [8]. Table rows refer to a different open/close configuration of the panels, while columns to a different frequency band, identified with its central frequency. Remembering that frequencies double from one octave to the next one, "central frequency" implies that the lower and upper bounds f_l and f_u of a band are related to its central frequency f_c through

$$f_c = \sqrt{f_l \cdot f_u} \tag{1.1}$$

which means that the central frequency of a band is the geometrical average between the band bounds.

In "CLOSED" and "OPEN" configurations, all panels are in vertical position and at maximum aperture, respectively. Note that single-sector 2P was not considered alone as a panel layout, so the final count of configurations is 11.

[8]Post-processing carried out with software SAMURAI by SINUS.

Configuration	125 Hz	250 Hz	500 Hz	1000 Hz	2000 Hz	4000 Hz
CLOSED	2.09	2.02	1.90	1.95	1.92	1.69
0C	2.07	2.00	1.89	1.94	1.91	1.69
1F	2.07	2.00	1.89	1.92	1.90	1.68
1C	2.06	1.99	1.88	1.91	1.89	1.67
1FC	2.06	1.99	1.88	1.88	1.88	1.66
1P	2.06	1.99	1.88	1.92	1.90	1.68
1FCP	2.03	1.96	1.85	1.86	1.85	1.65
2F	2.06	1.99	1.88	1.93	1.91	1.69
2C	2.05	1.98	1.87	1.91	1.89	1.68
2FCP	2.02	1.95	1.84	1.88	1.86	1.67
OPEN	1.96	1.89	1.78	1.81	1.81	1.62

Table 1.1: Global reverberation times ($T30$ [s]) of the theater's main hall for each considered panel open/close configuration, divided in octave bands.

2 | BIM context for a theater-Digital Twin interface

The possibility of integrating real-time data, that are going to be introduced later in section 3.1.1 on page 21, with a digital representation of the Roberto de Silva theater's inner space, collocating it in an appropriate Building Information Modeling system of processes, is the whole point of the project which the present thesis is part of. This chapter is dedicated to introducing the BIM paradigm and to viewing this work from its perspective.

2.1. The BIM approach

As a matter of fact, Building Information Modeling is a work approach. The idea of taking advantage of then-in-development Computer-Aided Design (CAD) software to include a building's functional and performance characteristics and go beyond mere structural design is far from recent, originating in the 1960s and 70s [4][13]. According to [11], "BIM is a process of maintaining a repository of all the information relevant to a building or construction project throughout the different phases of the project lifecycle [...] This information can be used in combination or separately, but not in isolation, in the sense that it will always be subject to some integrity and cross-checking". In other words, it is a framework that augments, using dedicated software[1], the traditional, static design of a physical structure through all kind of concerning digital assets and processes, to include all of its phases' management, even usage planning, transformation and discontinuation/demolition. These processes are shared between involved stakeholders and technological systems, facilitating real-time collaboration and making it especially useful in Facility Management contexts. The main advantages of conforming to BIM comprehend error and construction material waste reduction, making a project self-aware of collisions beforehand, and money and time optimization, with consequent savings of energy and resources in general

[1]The most popular one to create 3D models for BIM is Revit, developed expressly for Building Information Modeling methodologies in the late 90s and bought by Autodesk in 2002.

[11][12].

One concept and tool oftentimes associated with BIM is the Digital Twin (DT), a dynamic, data-driven virtual replica of a system, commonly implicating a physical structure, supporting its analysis and simulations inside of it, based on the mathematical models employed to describe the system itself. A bidirectional relationship exists between the real, physical structure and its digital twin, so that a status change in one of the two is reflected in the other one as well (a one-direction relationship, from the physical twin to the digital counterpart, would be rather called a "digital shadow"). [6]. DTs are crucial in remote monitoring: this is why a real-time information flow between physical and digital environments is contemplated in most of DT applications. This real-time flux is customarily actualized with the help of IoT devices and connected sensors that keep the digital twin status up to date.

An interesting pathway emerges with the combination of Digital Twin and Virtual Reality (VR) technologies. VR provides an immersive, simulated 3D environment, and the two together allow real-time visualization and manipulation of complex systems in an intuitive 3D space [9].

All of these notions joined together perfectly suit the necessity of taking multifunctional room management to the pace traced by industry 4.0, for which the literature coined the name "building acoustic modeling" (BAM) as the use of BIM in the building and room acoustics scope and marking a branch of the new acoustics automation field applied to acoustics in buildings [3][10].

Let us now try to assemble the concepts of this section in one project scheme idea, with the software we developed, called **Hallbridger**, being the communication channel in between physical (the real hall) and digital (its virtual representation) twins.

2.2. Hallbridger in the BIM context

Although it is not the only possible way to realize the user interaction with the 3D model of the theater inside a BIM process, the idea exposed in [3] encompasses the employment of a digital twin, which turns particularly useful in our case study. Indeed, an ideal implementation of the system of fig. 1 on page xii would involve a virtual reality environment, specializing the scheme in one like in fig. 2.1 on page 18. In a context like our polyfunctional hall BIM system, the information conveyed by the sensors in the room would be physical and atmospheric data that affect the room acoustics: temperature, humidity, air density and pressure. Pressure sensors installed in the seats could count the number of people in the audience, crossing data with the ticket sales software, whereas microphones and sound level meters could record acoustic signals to derive, among others, ambient background noise, reverberation times and acoustic power distribution [3]. The data coming from the 3D model, in addition to the visualization system, are fed to an internal piece of software that performs acoustic simulations on the fly to output an "acoustic snapshot" of the current 3D hall (which mirrors the real one, at this point). This snapshot could consist of a set of acoustic descriptors such as Definition (D_{50}), Clarity (C_{80}) and Early Lateral Energy Fraction (LF) indices, whose values should be visible concurrently with the 3D hall. Users and customers are supposed to view the hall twin (ideally employing a VR headset), operate in the virtual environment ("configuration modification" block), listen to the resulting acoustics ("acoustic tests" block, that exploits the acoustic simulation software too) using music or voice samples as playback signals[2]. The reverse path makes sure the virtual hall is not limited to a digital shadow of the real one. The scheme introduces the name "Hallbridger", that we mentioned in the previous paragraph, for the "ad-hoc software interface" between the FM software and the 3D model file that is the object of this thesis' development activity.

On top of all of this, the fact that such a system would benefit from the opportunities offered by BIM entails a number of advantages, like the possibility of automating the hall preparation for usage cases that might be deemed as "standard" by the facility manager, not requiring special attention on the acoustics by operators and musicians. Supposing the venue and its virtual environment are remotely linked through an internet connection, and a technician is allowed to control more than one hall, they could manage their staging from one place without having to be present in any of them. Apart from the specificity of multipurpose rooms, of course the system would be able to help optimizing maintenance processes, analyzing the system's status and applying prediction algorithms, and warn about to-be-taken actions in real time.

[2]Ideally, positioning both source and listening spot arbitrarily in the virtual space.

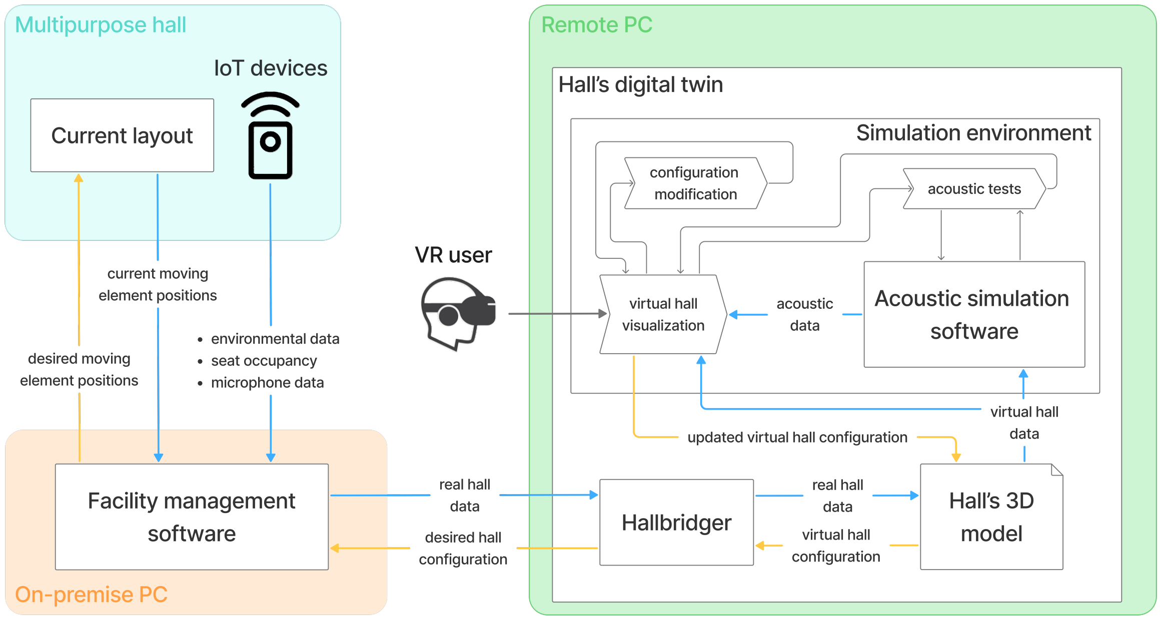


Figure 2.1: Proposed scheme of integration between the real hall and its digital twin for virtual reality utilization.

2.3. System implemented by Hallbridger version 1.0

So far we formulated problem and solution broadly speaking, thinking about a final, ideal version of the software system of the project. Hallbridger 1.0 focuses on just a first connection between CURIO © and a model of the theater hall conceived to fit in a BIM context. In line with what has been said so far, this connection is bidirectional, so the real-time data coming from the theater are read by the software and used to modify the model file, and data can be exported at any time from the 3D model back to CURIO to be read by it, as depicted in figure 2.2. More in detail:

1. Hallbridger receives real hall data from CURIO[3]. A modular design was envisioned for the communication between the theater and Hallbridger, so that the theater's facility management system is encapsulated and the software's code doesn't really depend on it - or, better, so that the software is easily extendable to adapt to any stage management software, as specified in section 3.5.2 on page 45.
2. Along with the data from CURIO, the Hallbridger interface receives as input the 3D model too, to view the position of movable elements contained in both input

[3]This is so far the only information that Hallbridger receives from the real hall: no IoT device is present in the theater, nor any other data retrieval system is employed to collect further information.

files and let the user compare values.

3. After processing the data coming from CURIO and the digital model of the theater, Hallbridger overwrites the data in the 3D model with those from the real hall.

4. Finally, it is possible for Hallbridger to extract the data inside the digital model and put them at CURIO's disposal, following the same data structure used by CURIO, so that they can be fed as input to CURIO and be used to command element repositioning in the real hall[4].

A real, complete BIM system implementation, or even any strategy to run acoustic simulations on the digital model of the hall, is out of this thesis's scope. Therefore, even if not required a priori, Hallbridger was "enriched" with functionalities that pertain the "Virtual hall" block of 1: it has a dedicated, user-interactive part of the graphical interface for the 3D viewer ("virtual hall visualization" in that scheme), a modification mechanism to edit moving element positions in the 3D hall ("configuration modification"), and a simple hall configuration matching system to identify the hall layout represented in the 3D model and highlight the corresponding global RT[5] ("acoustics evaluations").

Note that all three system components (CURIO, Hallbridger and the 3D model) are present on the same machine (theater PC): this is better explained in the next chapter, at section 3.3.2 on page 29.

[4]CURIO is not capable of importing external files at the moment: it would be up to its development team to furnish it with such a feature, so this part of the connection is not foreseen in the present work.

[5]Because both the measurements and the project focus on this major acoustic index, as stated in section 1.3 on page 10 and in [3], acoustic information consists of reverberation times in the initial version of Hallbridger.

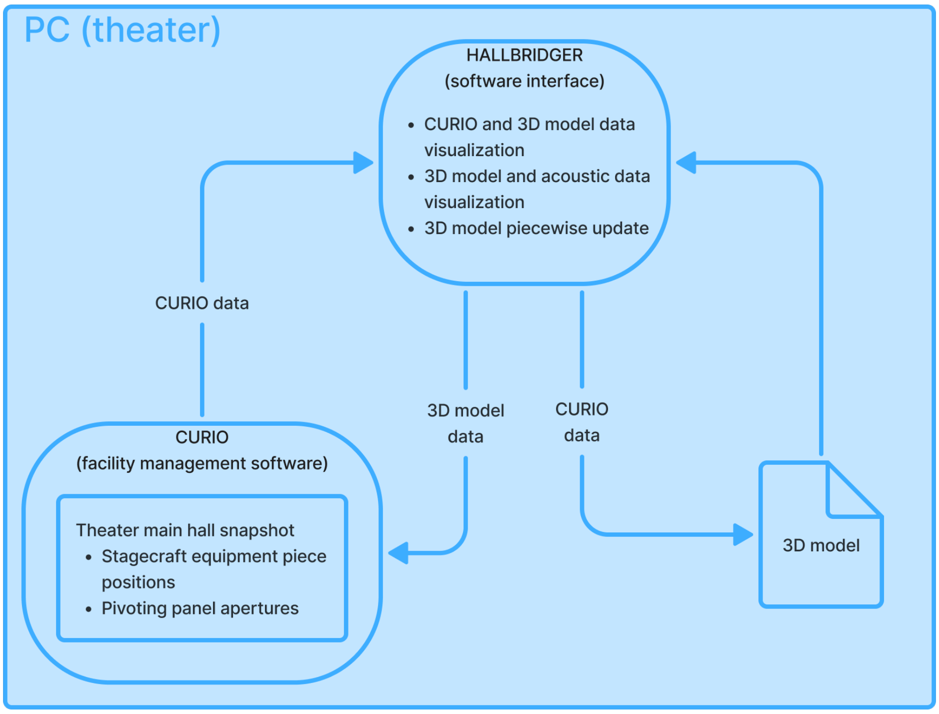


Figure 2.2: Scheme of the CURIO-3D model interface system for the software's first version.

3 | Software development

This chapter constitutes the documentation for Hallbridger's software that realizes the interface between CURIO and the digital 3D model of the hall and was developed for this thesis project. The diagrams were created using draw.io, and following UML standards.

3.1. Preface: input files

First of all, some preliminary considerations must be made about the files that the interface accepts as input. As we saw in 2.2, there is one input file for each of the two parts that Hallbridger put in communication: the file from CURIO, with the real hall's data, and the file containing the 3D model of the hall, representing the virtual counterpart of the former.

3.1.1. File from CURIO System © by *Decima 1948*

CURIO System is a stage control software adopted in various buildings all around the world, come to its 4.0 release at the time of writing. What is important to know about it as far as this thesis is concerned is its customizability. In fact, a first step into the development of Hallbridger was making CURIO capable of outputting a file containing information about the current position of the various elements that are connected to engines to move[1]. With reference to the theater in question, this is equivalent to

- 26 left wall pivoting panel array groups;
- 26 right wall pivoting panel array groups;
- 13 pieces of stagecraft equipment and movable architectural elements.

Since the model, for the time being, shows some of the pivoting panels only (section 3.1.2 on the next page), the information about stagecraft equipment positions is not utilized in the first release of Hallbridger, but it is ready to be the moment the 3D model of the

[1]This export function we need was not already present in CURIO, whose development team kindly accepted to help me. We did the job on April 7$^{\text{th}}$, 2025, as preliminary step in the thesis work.

hall includes the corresponding 3D elements in it. The tailored export feature that was implemented in CURIO for the purpose of Hallbridger consisted of a button added to the main panel of its user interface, which triggers a "snapshot" of the theater room, i.e. composing a file having a fixed defined structure with the data previously listed. This snapshot could be chosen to consist of a text file or a spreadsheet in XLS format. Since this output file does not have to be read from the final user and is only fed to the interface, and because strongly structured data are not necessary at this point of the data flow, a simple text file, UTF-8 encoded, has been chosen rather than an Excel file. The structure it follows is outlined in the `readme.md` file of the program, as mentioned in the corresponding section 3.5.4 on page 48, and is hard-coded in Hallbridger's code (see section 3.5.2 on page 45), meaning that it is not flexible and cannot be changed by the final user. This text file is then saved locally on the PC where CURIO is installed for Hallbridger to safely load and process it. The format of moving elements' identifiers is "XX.###", where

- "XX" is an alphabetical identifier for the moving element (e.g., "PS" for left wall panels and "PD" for right wall panels)
- "###" is a numerical identifier for the moving element

All the numerical data coming out of CURIO are integer numbers expressing measures in millimeters. For stagecraft equipment, it corresponds to the vertical distance of the piece of equipment starting from the top, so zero corresponds to the upmost position (completely rolled cable); instead, for pivoting panels, it is the stroke of the actuator that rotates the hinges to which the panels are hung (section 1.1.2 on page 5), so zero corresponds to a close panel, in vertical position. CURIO lets each custom installation of its software set the maximum allowed value for each of the elements it controls, so for each panel array grouping in our case. Anyway, the numbers are always bounded from 0 to 99999. For the moment, since CURIO "sees" and reads a single value for each control unit (i.e., for each actuator), the software has been developed and the 3D model has been created neglecting the fact that a differentiation in panel aperture within an individual panel array exists (see 1.1.2 and), and all panels belonging to the same grouping are shown with the same absolute aperture in the model.

3.1.2. 3D model file

A file containing the 3D model and supposed to be integrated in a BIM scheme may be implemented with a proprietary file format, primarily RVT for a Revit file, or an open one such as IFC (Industry Foundation Classes) by the international organization buildingSMART[2]. Picking one or the other influences the software design: in fact, Revit does not allow the modification of an .rvt file through its own API out of a Revit environment itself, which means that a Revit-add-on must be developed with which Revit's API can be used to edit the file, so Revit must be installed in the operating PC and be executing while using the add-on. This is the reason why `.ifc` file extension was preferred in the end: free and open source tools are made available for anyone to use, letting a developer integrate them into any supported software development framework. Most popular CAD modeling software for Building Information Modeling allows project imports and exports from and to IFC format, making it versatile and easily obtainable. Revit was the designated software to obtain the IFC file needed by Hallbridger. For the sake of this first version of the software, it was decided to consider the rotating panels of the theater only: modeling stagecraft equipment would require to build an IFC file from scratch, with a considerable amount of work implied, which is typically out of the bounds of a master of science thesis. Instead, limiting ourselves to the panels, we can afford to start from an existing digital model of the hall as "basis" for the IFC file, and then add the panels to it. The first phase of this process consisted in creating the Revit families (.rfa files) for the panels. Nested families are common practice when modeling in Revit to gain control over the atomic building elements: therefore, this approach was adopted for the panel column definition. Different families were specified:

- One for a single panel (an instance of type `IIfcFurnishingElement`[3]), called "SINGLE PANEL" and obtained through a simple extrusion of a rectangular surface
- One for a single panel (instance of type `IIfcFurnishingElement`), called "PANEL ROTATION", created with a SINGLE PANEL as nested family and characterized by a `IIfcPropertySingleValue` property named "PANEL ANGLE" falling under the `IIfcPropertySet` "Quote"
- One for a panel array of 5 panels (an instance of type `IIfcBuildingElementProxy`), called "PANEL BLOCK", and relative homonymous object type "PANEL BLOCK", encapsulating objects of type "PANEL ROTATION"

After that, the property PANEL ANGLE was made a shared parameter, rather than a property of the individual family, by loading into Revit an appropriate external TXT file filled with shared parameters, and loading the .rfa file of the PANEL ROTATION family on top of that: this way, the property remains visible from the upper-level family even if

[2]Actually, the IFC file format was invented to portray a structure in a fixed, immutable state, and not to be dynamically altered: we are a bit bending the rules for our purposes.

[3]This kind of IFC element was picked to better specify the panel, rather than leaving it to the default, generic `IIfcBuildingElementProxy`

the family PANEL ROTATION is later nested inside of PANEL BLOCK. From here, a first IFC file was extrapolated, opting for a simple export with default settings. The next step was to start building the final IFC file employing a static model of the theater. A SketchUp model (.skp) of the hall was imported into Revit to give the base 3D model on which "heaping up" the novelty elements, i.e. the panels. Then, 6 objects of type PANEL BLOCK were added to the model, precisely on the lateral wall of the parterre, so that the panel arrays belonging to sector 0C are modeled. The final step was to export an IFC file out of the resulting 3D model.

3.2. Purpose

Hallbridger software is the initial step of a large project potentially involving digital twins, BIM processes, acoustic simulations, virtual reality and Internet of Things. We assessed it as a necessary starting point for all the specific parts of the system, even though it does not implement a DT itself yet, or realize a proper BIM process, or perform numerical calculus. What is important in this phase is our interface being compliant with all foreseen possible future developments, that involve the fields we just mentioned.

It must be easy to make it part of a BIM process. The final goal is to have the theater hall maintenance automatable and its acoustic control potential fully exploited: this is perfectly achieved by adopting a Building Information Modeling approach, so in the end Hallbridger should be part of a BIM process. More specifically, for the reasons expressed in section 3.1.2 on page 22, the software must be able to handle an IFC file.

CURIO © is the facility management system involved in our case study, so, for the time being, communication with the real hall must work at least with that software. This implies that, at the moment, another goal is to be able to use the interface in any space where CURIO is employed to control its moving elements. In particular, motivated as in section 3.1.1 on page 21, the interface must accept extension `.txt` for real hall data.

Nevertheless, with future perspective, a general case in which other stagecraft control systems can be integrated with Hallbridger must be envisioned. Therefore, these further integrations must be made attainable in the simplest possible way.

The software interface, at least in its first version, must consist of a simple window that gives the possibility to view the data loaded from the two files (the text file coming from CURIO with the current configuration of the real hall and the IFC file containing the 3D model of the theater) and to update the data in the IFC file about stagecraft equipment positions and pivoting panel apertures, substituting them with those of the current real hall layout and refreshing all the views in the graphical interface. With the goal scenario in mind, where an operator tests the acoustics of the virtual hall and confirms its configuration, Hallbridger must provide an export function to extract the current 3D hall data and create a file containing them, too.

Another view the program must provide consists of the currently loaded 3D model of the hall and a table with reverberation times exposed in section 1.3 on page 10. If one of the hall configuration listed in this grid is recognized in the 3D hall currently loaded, the corresponding row must be highlighted, so that the operator knows what the average acoustic characteristics are expected from the hall represented by the 3D model.

Finally, the product must ensure the possibility of establishing an automatic data flux for its usage in real time: once it is connected with an installation of CURIO and with a virtual model of the same hall, every change in the real hall is automatically detected, read and viewed in the interface, and the 3D model is updated so that it mirrors the real situation both in the data grids and in the 3D viewer.

Enumerating them, the application goals are then

- [G.1] Must be ready to be adopted by a BIM system through an IFC file
- [G.2] Must be able to connect to CURIO System © through a textual file
- [G.3] Integration with facility management system must be extendable to software other than CURIO System ©
- [G.4] Must give the possibility to load real hall data coming from the FM software and show them in the interface
- [G.5] Must give the possibility to load the 3D model of the hall and show its data and the 3D hall in the interface
- [G.6] Must give the possibility to overwrite the data of the 3D model with those coming from the real hall and show the updated data and 3D model
- [G.7] Must give the possibility to export the data currently in the 3D model for the FM software to read them
- [G.8] Must give the possibility to view the 3D model of the hall and the acoustic information relative to it
- [G.9] Must give the possibility to create an automatic real-time data flow from the real hall to the 3D one

3.3. Software requirements specification

3.3.1. Functional requirements

For each of the defined goals, the relative software requirements are inferred, which define application development design choices.

- [G.1] **Must be ready to be adopted by a BIM system through an IFC file**
  - [R.1] The software must accept extension `.ifc` for the file containing the 3D hall
  - [R.2] The software must be able to handle an IFC file for the 3D model
- [G.2] **Must be able to connect to CURIO System © through a textual file**
  - [R.3] The software must accept extension `.txt` for the file containing real hall data
  - [R.4] The software must be able to read data coming from CURIO
  - [R.5] The software must be able to write data on a file that can be read by CURIO
- [G.3] **Integration with facility management system must be extendable to software other than CURIO System ©**
  - [R.6] Communication with the facility management system must occur via a custom API
- [G.4] **Must give the possibility to load real hall data coming from the FM software and show them in the interface**
  - [R.3] The software must accept extension `.txt` for the file containing real hall data
  - [R.4] The software must be able to read data coming from CURIO
  - [R.7] The user must be allowed to select the input file with real hall data
  - [R.8] The software must display the loaded data
- [G.5] **Must give the possibility to load the 3D model of the hall and show its data and the 3D hall in the interface**

  - [R.1] The software must accept extension `.ifc` for the file containing the 3D hall
  - [R.2] The software must be able to handle an IFC file for the 3D model
  - [R.8] The software must display the loaded data
  - [R.9] The user must be allowed to select the input file with the 3D hall
  - [R.10] The software must display the loaded 3D model

- [G.6] **Must give the possibility to overwrite the data of the 3D model with those coming from the real hall and show the updated data and 3D model**
  - [R.1] The software must accept extension `.ifc` for the file containing the 3D hall
  - [R.2] The software must be able to handle an IFC file for the 3D model
  - [R.3] The software must accept extension `.txt` for the file containing real hall data
  - [R.4] The software must be able to read data coming from CURIO
  - [R.7] The user must be allowed to select the input file with real hall data
  - [R.8] The software must display the loaded data
  - [R.9] The user must be allowed to select the input file with the 3D hall
  - [R.10] The software must display the loaded 3D model
- [G.7] **Must give the possibility to export the data currently in the 3D model for the FM software to read them**
  - [R.1] The software must accept extension `.ifc` for the file containing the 3D hall
  - [R.2] The software must be able to handle an IFC file for the 3D model
  - [R.5] The software must be able to write data on a file that can be read by CURIO
  - [R.9] The user must be allowed to select the input file with the 3D hall
- [G.8] **Must give the possibility to view the 3D model of the hall and the acoustic information relative to it**

  - [R.1] The software must accept extension `.ifc` for the file containing the 3D hall
  - [R.2] The software must be able to handle an IFC file for the 3D model
  - [R.9] The user must be allowed to select the input file with the 3D hall
  - [R.10] The software must display the loaded 3D model
  - [R.11] The software must highlight the reverberation times relative to the current 3D hall configuration

- [G.9] **Must give the possibility to create an automatic real-time data flow from the real hall to the 3D one**
  - [R.1] The software must accept extension `.ifc` for the file containing the 3D hall
  - [R.2] The software must be able to handle an IFC file for the 3D model
  - [R.3] The software must accept extension `.txt` for the file containing real hall data
  - [R.4] The software must be able to read data coming from CURIO
  - [R.8] The software must display the loaded data
  - [R.10] The software must display the loaded 3D model
  - [R.12] The user must be able to activate automatic reading of real hall data file
  - [R.13] The user must be able to activate automatic reading of the 3D hall file
  - [R.14] The user must be able to activate automatic overwriting of the 3D hall data with those coming from the real hall

In summary, the functional requirements for the program to meet can be listed as follows:

- [R.1] The software must accept extension `.ifc` for the file containing the 3D hall
- [R.2] The software must be able to handle an IFC file for the 3D model
- [R.3] The software must accept extension `.txt` for the file containing real hall data
- [R.4] The software must be able to read data coming from CURIO
- [R.5] The software must be able to write data on a file that can be read by CURIO

- [R.6] Communication with the facility management system must occur via a custom API
- [R.7] The user must be allowed to select the input file with real hall data
- [R.8] The software must display the loaded data
- [R.9] The user must be allowed to select the input file with the 3D hall
- [R.10] The software must display the loaded 3D model
- [R.11] The software must highlight the reverberation times relative to the current 3D hall configuration
- [R.12] The user must be able to activate automatic reading of real hall data file
- [R.13] The user must be able to activate automatic reading of the 3D hall file
- [R.14] The user must be able to activate automatic overwriting of the 3D hall data with those coming from the real hall

3.3.2. Non-functional requirements

For privacy and security reasons related to sensitive data about the theater management system, the computer on which the software should be installed is kept offline and accessible only from the internal network of the theater. In fact, this is the same calculator where CURIO is installed and run and from which the movable elements in the hall can be controlled. Thus, all input files are going to be physically present on that same computer.

3.3.3. Technical constraints

The program is going to be installed and executed on a Microsoft Windows system, precisely version 10 Enterprise 2019 Edition. This is the only system requirement since the program is going to work offline with files locally present on the computer of the theater and the operations it performs are quite basic, so any desktop hardware configuration supporting a modern operating system like Windows 10 will suffice for running the developed software.

3.4. Use cases

The first version of Hallbridger presents the use cases in diagram fig. 3.1 on page 31, listed and detailed below. Four of them (two for manual/automatic file upload and two for real/3D hall file) are about input file loading. Manual file upload is performed by the operator, while the automatic uploads are by the system. All of these upload operations embed the visualization of the uploaded data, and the 3D model if this is the case. The user can also update the 3D model of the hall and view the updated data and model, or export its data to an output file readable by the theater software. They are also able to switch to a view where acoustic data of the hall are displayed, along with the 3D model, where the various 3D elements can be selected to view and edit their properties. Note that "Export 3D hall data" is a manual operation only, performed by the operator, and doesn't have a relative automatic action performed by the system: this comes in accordance with the final scenario of the project, in which the theater operator, after modifying the virtual hall configuration and trying out its acoustics, wants to confirm the current configuration of the virtual hall and export it to the real hall pressing a button. Other actions the user may perform regard configuration settings about data and 3D model visualization, and about automatic input file actions.

The description of three main use cases follows (the ones relative to manual upload and update of input files), each with corresponding sequence diagram specifying the essential actions performed by the parties involved. "ref" blocks are used to reference other sequence diagrams (auxiliary sequence diagrams), for operations in common among more use cases. For document readability reasons, these, together with all the other use cases' description and sequence diagram[4], can be found at the end of this document, in the A, at A.2 and A.1 respectively.

The boundary-control-entity (BCE) architectural pattern was generally adopted for symbolism of components in the sequence diagrams. In some cases, though, for simplicity of implementation, the current solution does not decouple the boundary and control functions, making them overlap for some components. For instance, an I/O Interface element like a dialog window, that should be defined as boundary, formally assumes the "control" role if it actually processes the data sourced from user interaction directly in the class that defines it and switches messages directly with entity components. About this, the external libraries used for interacting with the 3D model are represented by a neutral symbol, but they serve *de facto* as "control" elements, exchanging messages with a "boundary" on one side and with an "entity" on the other.

[4] We decided not to show the sequence diagram of some trivial use cases, that would be made up of just one or two messages in total.

Figure 3.1: Use-case diagram for the first version of Hallbridger software.

3.4.1. Manually load and view real hall data

- Description: the user chooses a file containing the information of the moving elements of the real hall, and the software reads the file, and loads and displays the content
- Preconditions:
  - Input file can be found in the computer's file system and is accessible by the software
  - Input file has been formatted as outlined in README file
  - Software initialization has completed
- Post-conditions:
  - The data coming from the real hall are correctly read by the software and visualized
  - [3D hall data previously loaded and automatic discrepancy highlighting turned on] Differences between values from the real hall and the 3D hall relative to the same movable elements are highlighted
- Basic flow:
  1. Operator triggers the "Load real hall data" function
  2. System shows windows with file explorer
  3. Operator navigates the file system and chooses file to load
  4. System reads selected file
  5. System fills out Hallbridger inner variables
  6. System displays the loaded data
  7. [3D hall data previously loaded and automatic discrepancy highlighting turned on] System highlights discrepancies with corresponding data from the 3D model
- Exception flow: file extension not supported
  1. Operator triggers the "Load real hall data" function
  2. System shows window with file explorer
  3. Operator navigates the file system and chooses file to upload with not supported extension

4. System shows message "File extension not supported"

- Exception flow: error while reading file

1. Operator triggers the "Load real hall data" function
2. System shows windows with file explorer
3. Operator navigates the file system and chooses file to upload
4. System shows message "Error while reading hall data" with error details

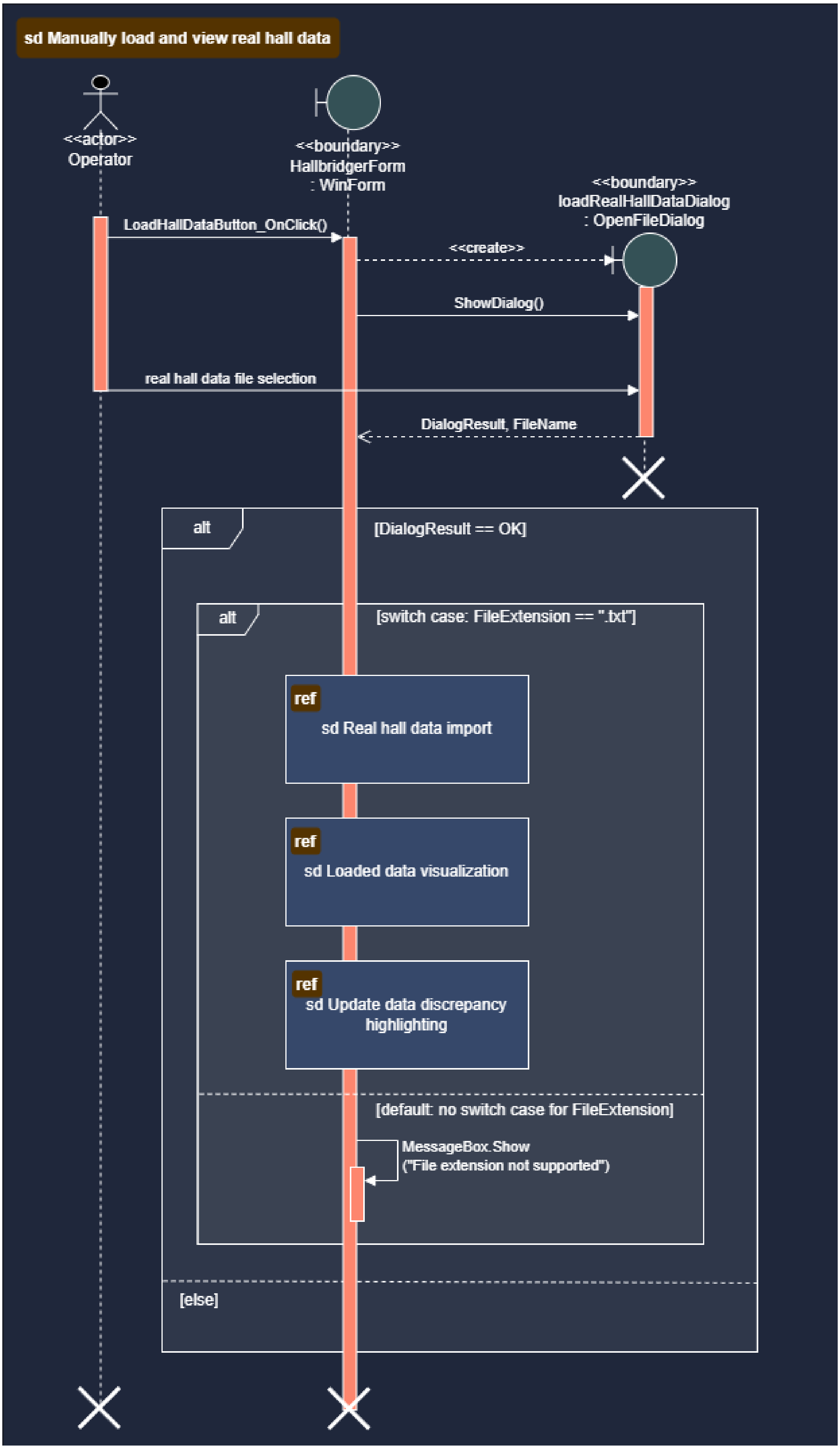


Figure 3.2: Sequence diagram for use case "Manually load and view real hall data".

3.4.2. Manually load and view 3D hall

- Description: the user chooses a file containing the 3D model of the hall; the software reads it and loads and displays the information about moving element positions; in addition, the user can view and explore the 3D model in a dedicated viewer, too
- Preconditions:
  - Input file can be found in the computer's file system and is accessible by the software
  - Input file has been formatted as outlined in README file
  - Software initialization has completed
- Post-conditions:
  - The data about moving element positions contained in the 3D model are correctly read by the software and visualized
  - The 3D model is visualized beside a table with reverberation times
  - [Real hall data previously loaded and automatic discrepancy highlighting turned on] Differences between values from the real hall and the 3D hall relative to the same movable elements are highlighted
  - [Moving element configuration in 3D hall recognized] Global reverberation times and elements of the 3D model relative to the 3D hall configuration are highlighted
- Basic flow:
  1. Operator triggers the "Load 3D hall" function
  2. System shows window with file explorer
  3. Operator navigates the file system and chooses file to load
  4. System reads selected file and fills out inner variables
  5. System displays the loaded data
  6. [Real hall data previously loaded and automatic discrepancy highlighting turned on] System highlights discrepancies with corresponding data from the real hall
  7. System displays the 3D model

8. [Moving element configuration in 3D hall recognized] System highlights global RT values and elements of the 3D model relative to the current 3D hall configuration

- Exception flow: file extension not supported

1. Operator triggers the "Load 3D hall" function
2. System shows windows with file explorer
3. Operator navigates the file system and chooses file to upload with not supported extension
4. System shows message "File extension not supported"

- Exception flow: error while reading file

1. Operator triggers the "Load 3D hall" function
2. System shows windows with file explorer
3. Operator navigates the file system and chooses file to upload
4. System shows message "Error while reading 3D model" with error details

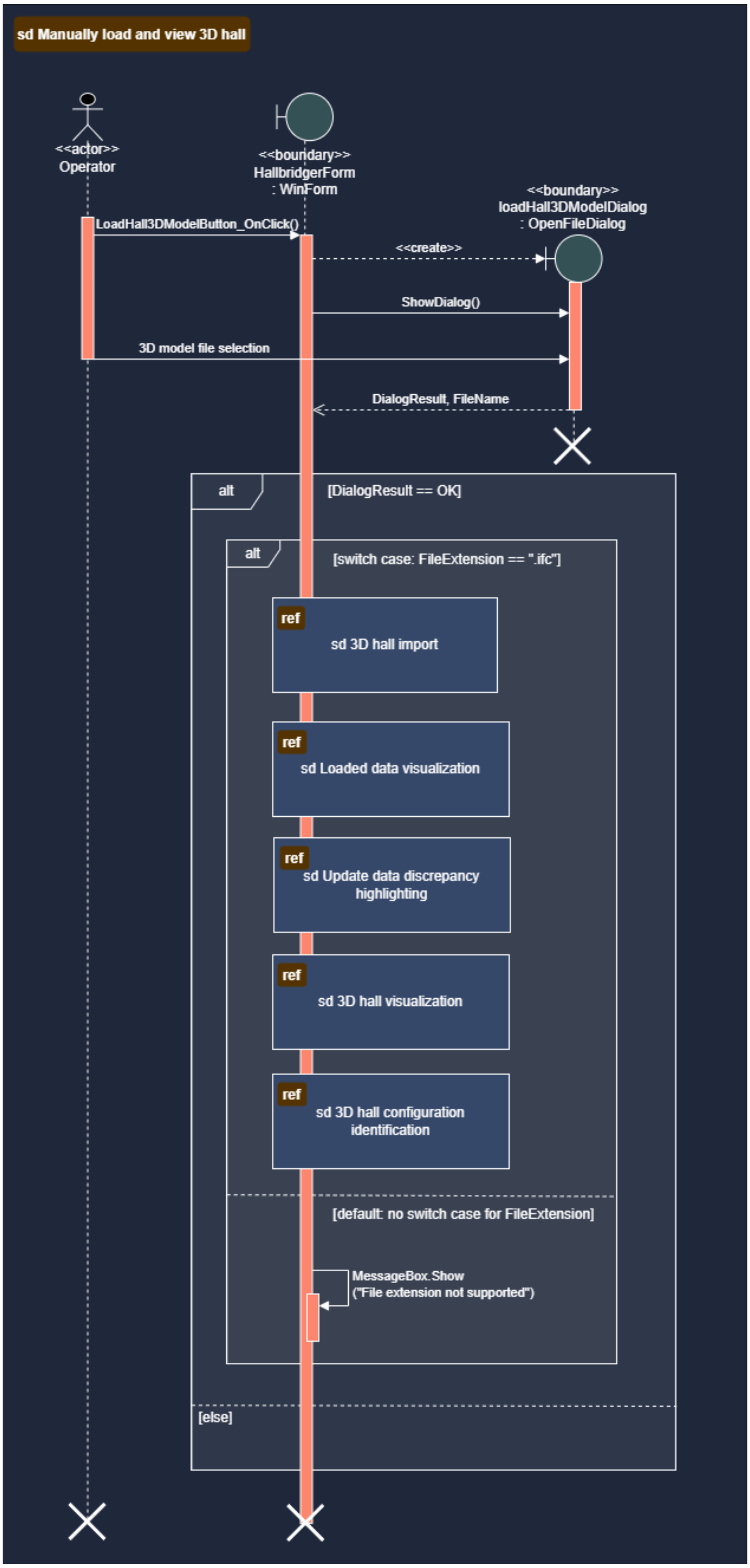


Figure 3.3: Sequence diagram for use case "Manually load and view 3D hall".

3.4.3. Manually update and view 3D hall

- Description: the previously loaded 3D hall file is modified overwriting the contained information about the moving element position with the previously loaded data coming from the real hall (if the 3D hall file hasn't been loaded yet, the user selects one from the file system); the software then reloads the file in the interface, displaying both the updated data and 3D model
- Preconditions:
  - Input file can be found in the computer's file system and is accessible by the software
  - Input file has been formatted as outlined in README file
  - At least some data coming from the real hall have been loaded and are shown in the corresponding tables of "Moving element data" tab
  - Software initialization has completed
- Post-conditions:
  - The data relative to moving element positions in the 3D hall match those relative to the corresponding moving element positions in the real hall
  - The 3D model of the hall is updated with the data coming from the real hall
  - [Automatic discrepancy highlighting turned on] No data discrepancy is shown for the items present in both the real and the 3D hall
  - [Updated moving element configuration in 3D hall recognized] Global reverberation times and elements of the 3D model relative to the updated 3D hall configuration are highlighted
- Basic flow:
  1. Operator triggers the "Update 3D hall" function
  2. [No 3D hall previously loaded]
     (a) System shows window with file explorer
     (b) Operator navigates the file system and chooses file to update
     (c) System reads file and fills out inner variables

3. System edits the loaded 3D hall file substituting values of movable element positions with corresponding ones previously loaded from real hall
4. System reloads the 3D hall file and updates inner variables
5. System displays the loaded data
6. [Automatic discrepancy highlighting turned on] System highlights remaining discrepancies between real hall and 3D hall (e.g., for element not loaded for one of the two halls)
7. System displays the updated 3D hall
8. [Moving element configuration in 3D hall recognized] System highlights global RT values and elements of the 3D model relative to the current 3D hall configuration

- Exception flow: no real hall data loaded
  1. Operator triggers the "Update 3D hall" function with no real hall data previously loaded
  2. System shows message "No real hall data loaded"
- Exception flow: file extension not supported
  1. Operator triggers the "Update 3D hall" function without having previously loaded a 3D hall
  2. System shows windows with file explorer
  3. Operator navigates the file system and chooses file to update with not supported extension
  4. System shows message "File extension not supported"
- Exception flow: error while editing file
  1. Operator triggers the "Update 3D hall" function
  2. System tries to edit file and runs into exception
  3. System shows message "Error while modifying 3D model" with error details

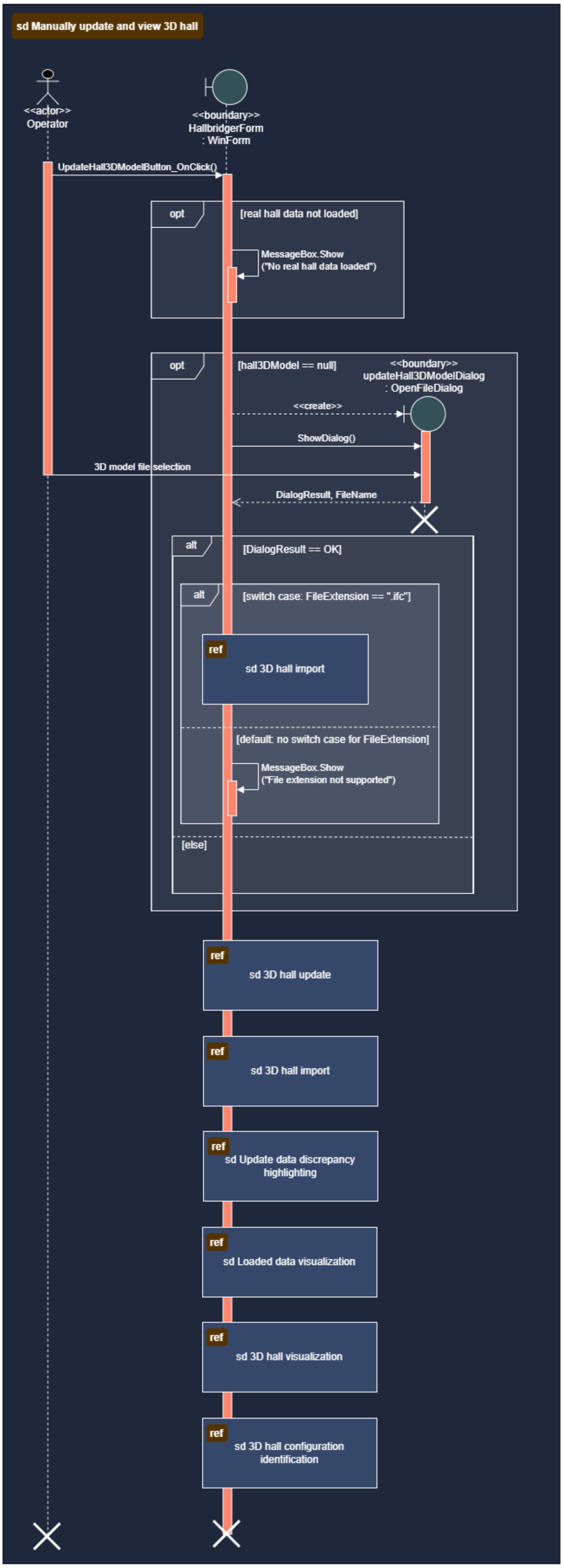


Figure 3.4: Sequence diagram for use case "Manually update and view 3D hall".

3.5. Design and development

3.5.1. Software development and documentation setting

Microsoft Visual Studio was chosen as Integrated Development Environment, as it is the worldwide most used and supported one for this kind of software, and C# as designated programming language, for its versatility, wideness of applicability and being well documented by Microsoft. One big advantage of using this language is that the open-source xbim toolkit - A .NET external library - can be adopted to "read, view and manipulate Building Information Models in the IFC format"[5]. Versioning was managed using GitHub (involved repositories can be found at https://github.com/jacopofantin/Hallbridger for the main project and at https://github.com/jacopofantin/Hallbridger_API for the API project).

3.5.2. Component diagram

The components in the running are essentially two: the Windows Form that implements the graphical interface, which is the main part of the software, and the API that allows for encapsulating the communication with the facility management software. Diagram 3.5 shows the two input files (the file with real hall data, read by means of the API, and the one with the hall 3D model, whose data are overwritten too), as well as the output file (the one with data exported from the 3D hall), as artifacts.

Graphical User Interface (GUI): Hallbridger form

A single window, implemented as Windows Form, showing the user interface of the program is enough to fulfill all the functional requirements about user-system interaction. It is implemented with a dedicated project in Visual Studio (.NET Framework 4.7.2), and in particular with the class HallbridgerForm, that is the main component of the entire software. The views of the window (the form) are subdivided into two tabs.

Moving element data tab The first tab of the form is dedicated to loaded data visualization. It consists of two groups of data grids: the upper one with data coming from CURIO, related to the real hall of the theater, and the lower part with data taken from the IFC model. Each group has three grids: one for stagecraft equipment positions, one with apertures of pivoting panel arrays on the left wall and one with apertures of

[5]source: https://docs.xbim.net/. GitHub repositories containg the code about the corresponding project can be found at [14]

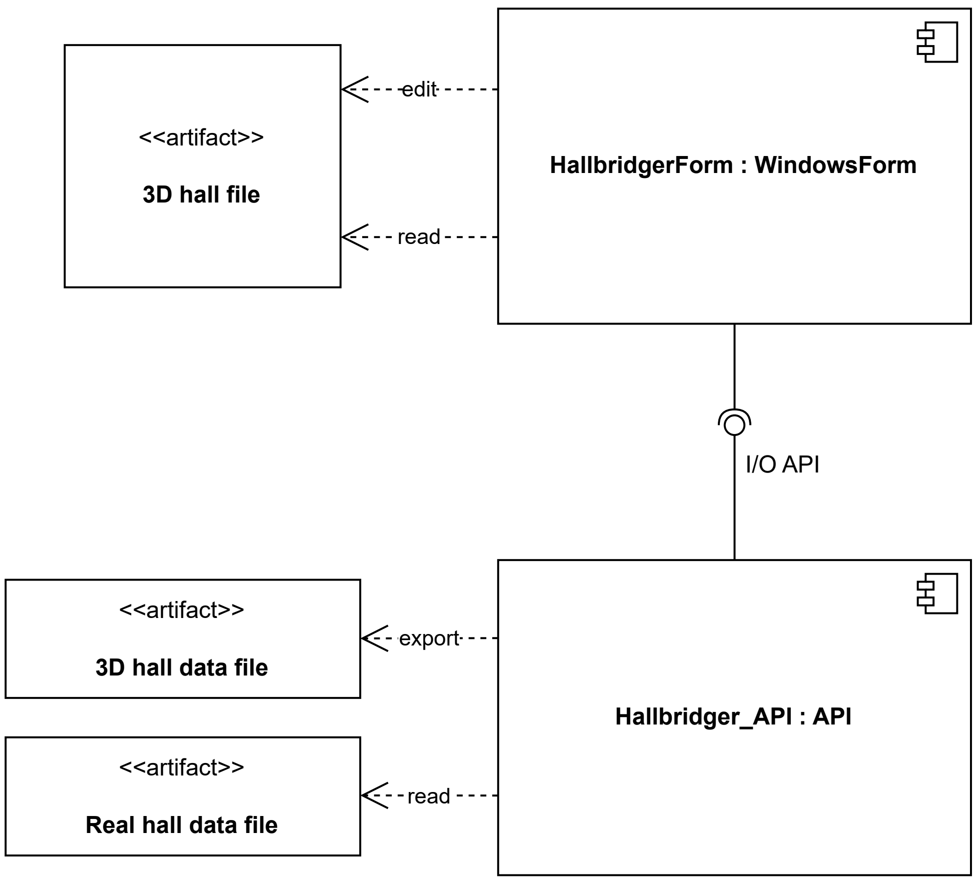


Figure 3.5: Component diagram for the Hallbridger app.

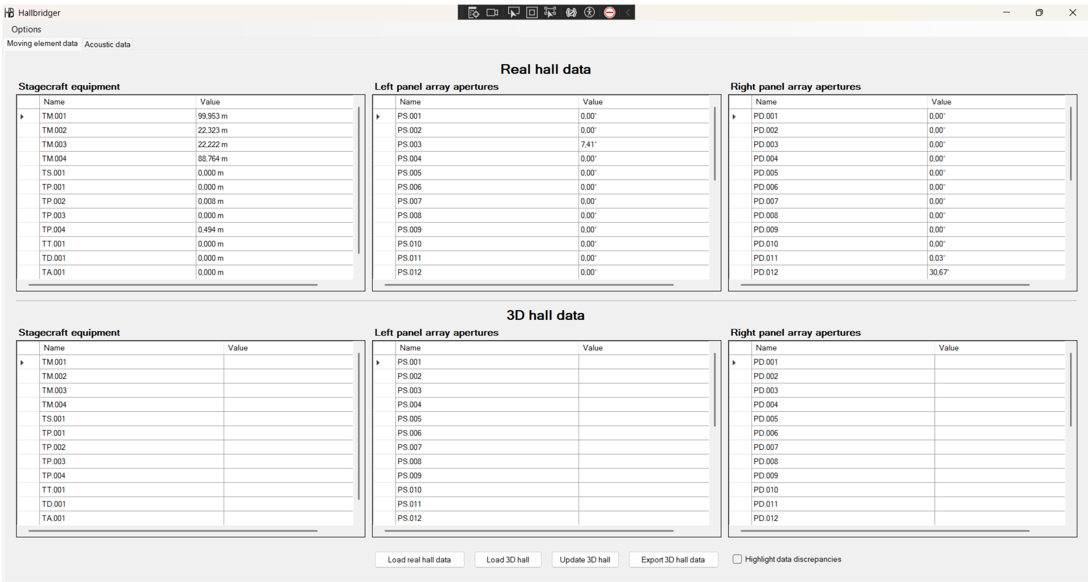


Figure 3.6: Hallbridger user interface with CURIO and 3D model data viewed in the "Moving element data" tab. Discrepancies highlighting is turned off.

those on the right wall. The various movable elements are identified the same way for CURIO source and IFC source, following the naming written in 3.1.1.

Next to each element identifier, the corresponding value appears, expressed in meters from its upmost position (for stagecraft equipment position) or decimal degrees for the rotation angle from the vertical position (for pivoting panels).

The bottom section of the tab shows four buttons: the first one ("Load real hall data") for uploading a text file coming from CURIO and showing its data in the corresponding tables (use case: 3.4.1); the second one ("Load 3D hall") for the same purpose with an IFC file (`.ifc`) (use case: 3.4.2); the third one ("Update 3D hall") to inject the values of the various elements coming from CURIO into the IFC model and refresh the values in the grids (use case: 3.4.3); the fourth one ("Export 3D hall data") to create a text file containing the data currently saved in the 3D model, using a structure analogous to the one of the input file coming from CURIO (use case: A.1.4). Next to the buttons, the "Highlight data discrepancy" checkbox allows the user to show or hide the highlighting, coloring their grid rows with the same color, of differences between values referred to the same moving element between the real hall and the 3D hall (use case: A.1.5).

Acoustic data tab The second tab is the core part of the program: it hosts a 3D viewer for the 3D model of the hall to be shown, and a table with global reverberation time values based on the acoustic measurements described in section 1.3 on page 10 (exactly table 1.1

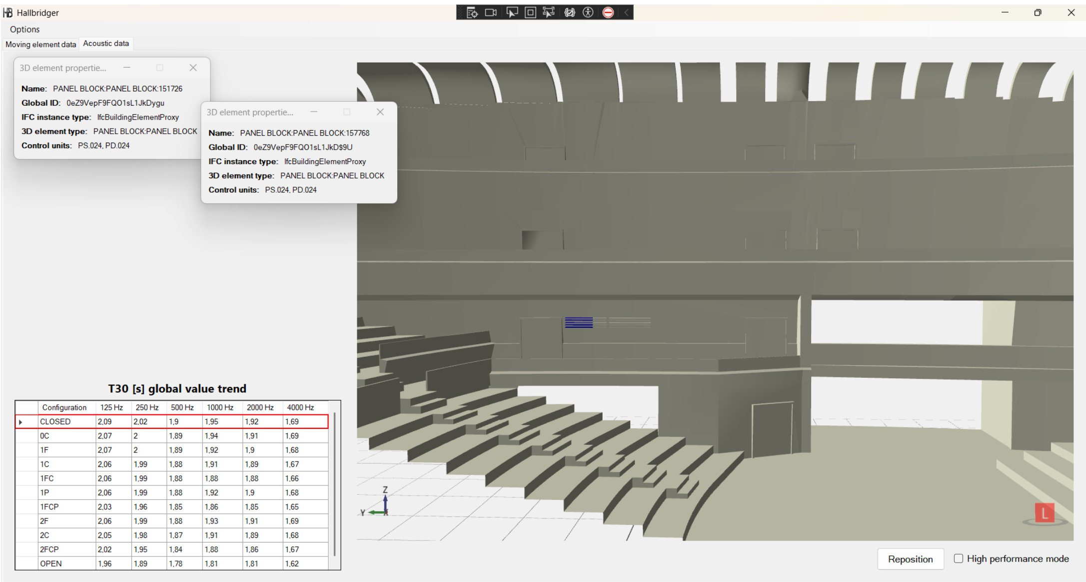


Figure 3.7: Hallbridger user interface with 3D hall and reverberation times viewed in the "Acoustic data tab". Panel configuration "CLOSED" is identified as the one represented in the loaded 3D model, and a couple panel arrays are selected in the viewer, so the relative 3D property windows appear on the left.

on page 13). The panel open/close configuration currently represented in the loaded 3D model, if it corresponds to a known one, is highlighted using a red border around the corresponding table row (use case A.1.7.

The top-left part of the tab is used by Hallbridger to open one popup window for each 3D element selected in the model, displaying its main properties (use case: appendix A.1.10 on page 76). Below the viewer, "Reposition" button resets the camera point of view in the viewer (use case A.1.8), and "High performance mode" checkbox switches between a lower resolution, higher performing 3D model representation, and the ordinary, more computing-power-demanding model visualization, with its original definition (use case A.1.9.

Menu bar Furthermore, Hallbridger is provided with a menu bar consisting of just one menu ("Options"). Its entries are

- Automatic discrepancy highlighting: toggle menu entry. Usage described by use case A.1.6
- Configurations: opens the Configuration window to see, change and save Hallbridger preferences about automatic input file usage (relative to use cases A.1.1, A.1.2 and A.1.3), namely

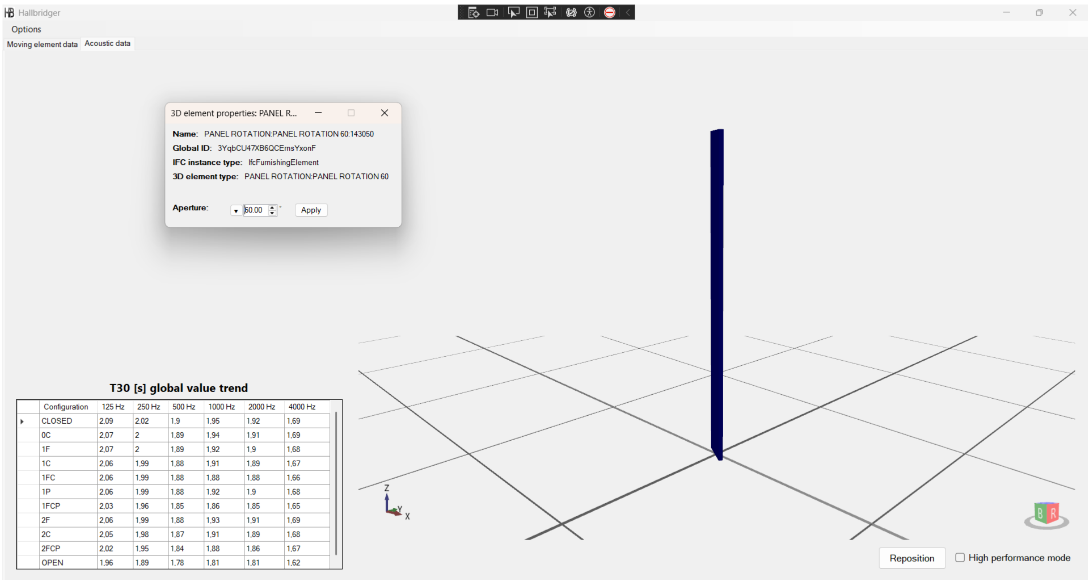


Figure 3.8: Hallbridger user interface with a single 3D panel in the "Acoustic data tab". The panel is recognized as a pivoting panel, so property "aperture" is shown with its value, editable by the user.

- File path: for both real and 3D hall input files, the file location and name so they can be automatically used by the interface
- Interval: for both real and 3D hall, how frequently the system should autonomously check for and use the input file
- Active: for both real and 3D hall, whether the automatic input file usage is enabled or not
- Operation: for 3D hall, whether the input file should be used to just load and view the data and the model ("Load"), or to be edited using the real hall data and to then view data and model ("Update")

The use case relative to the settings window is A.1.11.

Application Program Interface (API): Hallbridger API

A simple web API was set up to modularize the interaction between the real hall and Hallbridger. This solution makes Hallbridger's code independent from the software installed in the hall. However, for the first version of Hallbridger, since it has to connect to CURIO only, the modularization has been only partially implemented (see the existing routing later in in this section), and CURIO data interpretation is still largely hard-coded. Note that there is no need to employ an API for the interaction between Hallbridger and

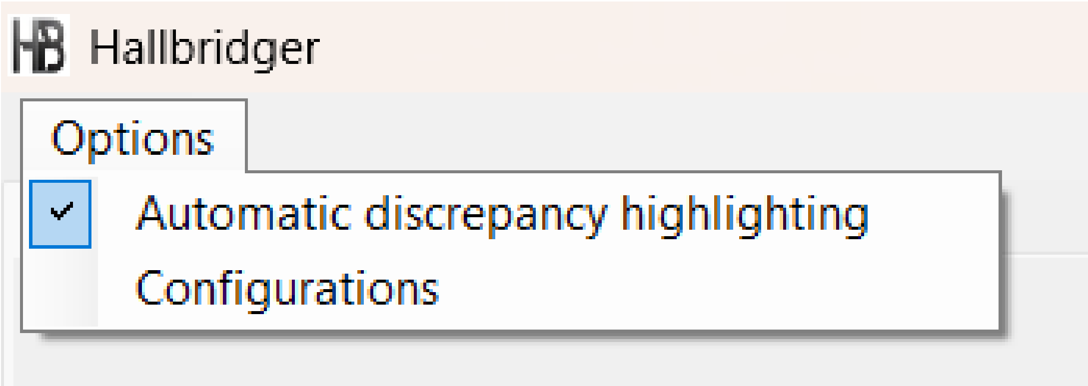


Figure 3.9: Menu bar of Hallbridger's user interface, with enabled option "Automatic discrepancy highlighting" (checked).

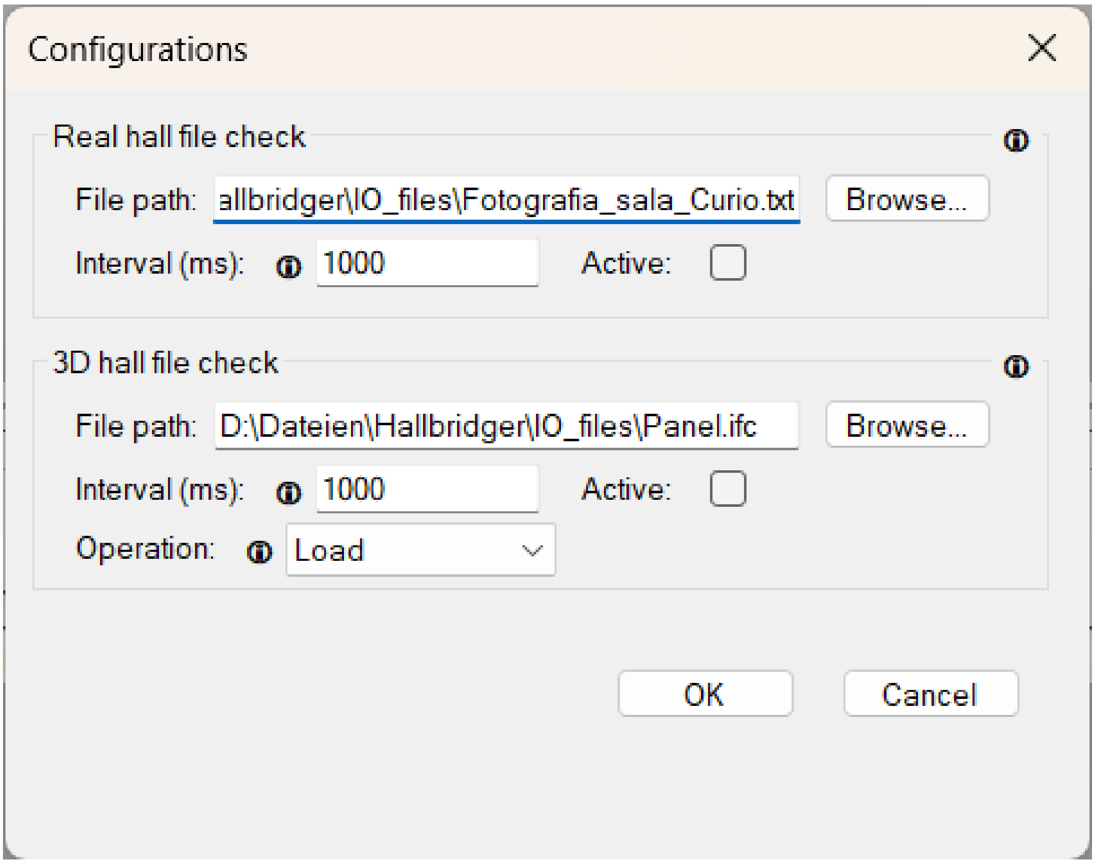


Figure 3.10: Hallbridger configuration window with default settings.

the 3D model, because we don't deal with another, external software for that part of the communication, but the user is responsible for directly putting an IFC file at disposal in the PC's file system[6]. Operatively, a dedicated project on Visual Studio was created. For highest possible compatibility and to best fit the main one, the project was chosen to use ASP.NET Web API framework in accordance to the main project's one (.NET Framework 4.7.2). The designated web server to run the API is IIS Express, installed on the same computer where the development took place: in fact, the main WinForm component calls the API through localhost, at port 44307 (https://localhost:44307/). This works until Hallbridger is installed and used on the same PC where the FM software runs, and not remotely (in which case, a remote connection must be established).

Swagger/OpenAPI documentation for the API endpoints was generated through the NuGet package *Swashbuckle* and is present, once the web server is running, at https://localhost:44307/ (Swagger UI) and at https://localhost:44307/swagger/docs/v1 (in raw JSON format).

The main parts of the API are described as follows:

- Controllers: the first routing for the web service calls, they define the operation that has to be performed
- Parsers/formatters: the second routing level defines which is the right class to process the input/output file based on its extension
- Models: classes that define a data model based on the FM software we are working with

Controllers The only controllers implemented so far are the one for importing data (ImportController.cs) and for exporting them (ExportController.cs). Both define one method for each file extension to parse (importing) or format (exporting) data contained in input or output files, respectively, having that extension. These methods call, in turn, the `Parse()` and `Format()` methods defined in the respective parsers or formatters.

Parsers/formatters One parser and one formatter for each file extension are necessary to define, respectively, all the `Parse()` and `Format()` methods that are called by, respectively, the import and export controllers. The former interpret input files to be imported, so they are conscious about their content and fill out a data structure that follows the

[6]For the various 3D elements of the model to be recognized, the associations between their identifiers (IFC's `GlobalId` property) and the corresponding real hall elements must be hard-coded in Hallbridger: this is another part that is not customizable by the user yet.

appropriate data model, that can be stored in Hallbridger's inner dictionaries directly. The latter work the other way round: they receive from Hallbridger the data structure filled with data from its inner dictionaries, and they build the output files' content that is then returned Hallbridger and used by it to generate the file in the desired location. For the scope of this thesis, the only managed file extension is `.txt`.

Models Based on the stagecraft control software we are connecting to (in our case, only CURIO System's), the right data model is adopted, that of course can differ from one FM software to another depending on how they implement the export function. Models define the data structure that is followed in Hallbridger to view the data and to correctly read and write input/output files.

Compatibility tests Not every project build could make the web services work. Indeed, a compatibility issue with the xbim library arose (in particular, with the variants of the geometry engine for topological operations and rendering of the 3D model, Xbim.Geometry.Engine32.dll and Xbim.Geometry.Engine64.dll), even though the API is not employed to interact with IFC files. To resolve the problem, a choice had to be made in compiling the code for x86 systems or x64, and the latter was preferred, for it is the most widely used nowadays.

3.5.3. Deployment diagram

To complete this software design part, the deployment diagram in fig. 3.11 on the facing page that resulted from this work on the first Hallbridger version was drawn. Consistently with what we said in section 3.3.2 on page 29, all involved software modules and files must be present on the same device, "Theater's PC". In fact, it serves as web server too, for the web services to run. These communicate via HTTP with Hallbridger main program, while connection with CURIO takes place offline: both `Real_hall_snapshot.txt` and `3D_hall_snapshot.txt` are written on the file system and read asynchronously. The same happens between Hallbridger and the file of the 3D model.

3.5.4. Instruction and configuration files

README file The README file - the .md file automatically generated by GitHub - was compiled with all relevant instructions for the final user to use the software. It gives information about the configuration file (see 3.5.4), and illustrates how input files must be structured so as to be correctly read by the program.

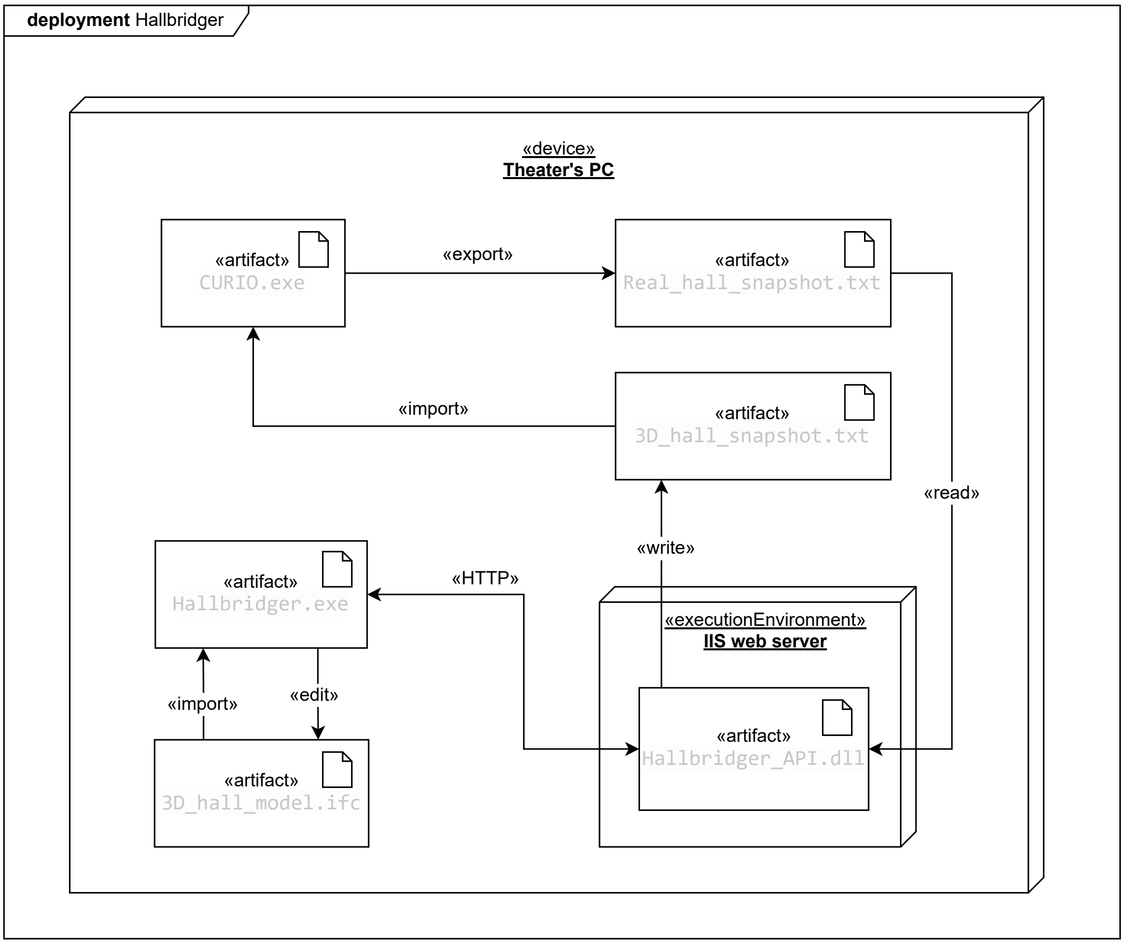


Figure 3.11: Deployment diagram for Hallbridger 1.0.

INI file A configuration file (.ini) was added to save the custom user preferences that were detailed at section 3.5.2 on page 44.

3.6. Tests

Tests couldn't be performed in the theater because of responsibility issues: the "Roberto de Silva" is a living venue, so there's no protected digital environment within it that can be employed as test environment to try the software out. Nevertheless, a sample 3D_HALL_SNAPSHOT was obtained importing a sample 3D model, modifying some of the aperture values for its pivoting panels and using Hallbridger's export function. The resulting text file was forwarded to *Decima 1948*, who could just confirm that the file was readable by CURIO, presenting the data in the same way it exports to.

4 | Conclusions and further work

This dissertation was produced to present the development of a software that was judged as indispensable to start working on the project discussed in [3]. The objective of completing a working application that can be installed and utilized in the theater in Rho is achieved, and the fact that we could not try it out on premises does not invalidate the result. It is rather a potential obstacle to interoperability that is intrinsic to the BIM paradigm for later software evolutions: some compromise between an open connectivity and a completely close and wired environment must be reached to enable the system to acquire data contributions from IoT devices and wireless sensors in general. Moreover, a solution that unlocks a remote connection between the FM system of the venue and the PC with Hallbridger and the tridimensional representation of the multipurpose hall needs to be agreed upon to let a human hall manager access the virtual hall environment and relative acoustic evaluation system without having to be on site, finally leveraging the capabilities of a digital model. For this purpose, possibilities that exploit the secure communication channel used by *Decima 1948*'s remote support system for their customers could be explored.

The system illustrated in 2.2 is a starting point scheme for implementing the final system we hypothesized, and has to be detailed and modified with specific software modules to make it really adopt BIM and use digital twins. A non-exhaustive list of future developments is compiled here.

BIM system implementation In practice, what is needed to call the context a BIM ecosystem can be divided in

- Structured non-geometric data: rich attribute data should be incorporated into the IFC model, such as acoustic specifications for the materials, acoustic performance metrics and maintenance schedules, to enable analysis and simulation later used by the DT, and lifecycle management.
- Standardized processes: a BIM Execution Plan (BEP) can assist with outlining roles of involved actors, material and software, Levels of Development (LOD), deliverables

and quality checks, while information requirements against our needs, and their automated validation, can be defined with the Information Delivery Specification (IDS) standard[1].

- collaborative workflows: a Common Data Environment (CDE) would instead facilitate versioning, federation of models and issue tracking via the BIM Collaboration Format (BCF) standard[1].

Besides, the DT system will have to be designed, with its distinct analysis software and all the technology stack that constitutes the communication channel with the physical twin. As stated in 2, we would also like IoT devices and a virtual reality environment connected to the twins. "Hallbridger" would then be the name assumed by this whole BIM system that uses a DT for the virtual side of the multipurpose hall.

Acoustics evaluation system improvement Hallbridger needs a way to properly estimate the room acoustics of the virtual hall. One solution is to extend the available RT value set by performing new measurements, covering more panel configurations, though a more comprehensive one would be to follow the indications in [3] to build acoustic simulations - calibrated with measured data - to estimate reverberation times for a discrete set of seat occupancy percentages, and an "acoustic procedure" to interpolate reverberation times at disposal and estimate RT values for arbitrary occupancy rates.

3D model enhancement For sure, the 3D model that we have for the time being must be finalized with all pivoting panels existing in real life. At that point, keeping into account different hall layouts (i.e., bringing stagecraft equipment positions in the 3D model and in the acoustic evaluation mechanism beside panel apertures) would refine the acoustics accuracy of the model and make the virtual environment more complete. Speaking of this, the representation of the hall could be more realistic if the aperture of each of the five panels in a panel array were studied more precisely in the Roberto de Silva theater, and if the way it is deducted from the value of its actuator stroke were discovered.

CURIO functionality expansion To have the FM software receive the data that "explains" how the moving elements in the facility should be laid out, this must be equipped with an appropriate "import" function, to be able to read data deposited by Hallbridger. To close the loop begun with the addition of its "export" feature, CURIO

[1] Another standard by buildingSMART, like IFC.

must now be enhanced with a way to load external data and change movable elements' position consequently.

Facility Management software encapsulation From the developed software side, the biggest improvement would be to make Hallbridger completely FM-software independent. Hallbridger API was created exactly for this purpose, but the data structure used by CURIO is to this day still hard-coded both for data storage in the program memory and for data interpretation in the `Parse()` and `Format()` methods of API parsers and formatters. It is necessary, and fairly easy by design, to expand the API adding a routing level on the controllers to manage the different FM software. It is self-evident that this affects the graphical interface too: a dropdown menu contains the available stagecraft control software, and data grids are shown accordingly after the choice is made. Continuing along the lines of freeing the software from hard-coded data, a setting-up procedure should be foreseen at program startup to allow the user manually correlate each moving element in the real hall with the corresponding one in the 3D hall.

Other future developments Instead, some immediate and evident improvements that could be made to Hallbridger for a next, not major release and that are to be considered optional include

- Unit of measurement atomically controllable by the user for each view where physical quantities are displayed;
- Connection to a database for acoustic data storage, most of all for reverberation times, both measured in advance and on the fly.

Bibliography

[1] M. Barron. *Auditorium Acoustics and Architectural Design*. Taylor & Francis, Abingdon, UK, 2009.

[2] M. Cairoli. The architectural acoustic design for a multipurpose auditorium: Le serre hall in the villa erba convention center. *Applied Acoustics*, 173, 2021.

[3] M. Cairoli and L. C. Tagliabue. Digital twin for acoustics and stage craft facility management in a multipurpose hall. *Acoustics*, 5:909–927, 2023.

[4] C. Eastman, D. Fisher, G. Lafue, J. Lividini, D. Stoker, and C. Yessios. An outline of the building description system. *Research Report No 50, Institute of Physical Planning, Carnegie-Mellon University, Pittsburgh*, 1974.

[5] I. O. for Standardization. Iso 3382-1, 2009. URL `https://www.iso.org/standard/40979.html`.

[6] S. Haag and R. Anderl. Digital twin – proof of concept. *Manufacturing Letters*, 15, part B:64–66, 2018.

[7] M. Hans and M. Joseph. Increasing liveness and clarity in a multipurpose civic center. *The Journal of the Acoustic Society of America*, 140, 2016.

[8] R. Orio. Pivoting panels as variable acoustic elements to optimise the reverberation time in the roberto de silva theatre. Master's thesis, Politecnico di Milano, Piazza Leonardo da Vinci 32, Milano, Italy, 2023.

[9] N. Saporiti, F. Strozzi, and T. Rossi. Digital twin relationship with virtual reality and augmented reality: a bibliometric review. *26th Summer School "Francesco Turco" Proceedings*, pages 1–7, 2021.

[10] J. Thede, N. Boyts, J. Teel, K. Butler, E. Bargas, and A. Roth. Building information modeling (bim) automation for architectural acoustics, mechanical noise calculation. *Journal of the Acoustical Society of America*, 151, 2022.

[11] J. Underwood and U. Isikdag. *The Handbook of Research on Building Information*

Modeling and Construction Informatics: Concepts and Technologies. Information Science Reference, 2009.

[12] G. van Nederveen and F. P. Tolman. Modelling multiple views on buildings. *Automation in Construction*, 1:215–224, 1992.

[13] J. K.-W. Wong, J. Ge, and S. X. He. Digitisation in facilities management: A literature review and future research directions. *Automation in Construction*, 92: 312–326, 2018.

[14] xBimTeam. xbim toolkit documentation, 2026. URL `https://github.com/xBimTeam`.

A | Appendix: Other use cases and auxiliary sequence diagrams

The use cases, with relative sequence diagrams, that were not included in the corresponding section of the main part of the document (3.4), are contained in this appendix, together with the auxiliary sequence diagrams that the various "ref" blocks refer to.

A.1. Other use cases

A.1.1. Automatically load and view real hall data

- Description: a timer gets triggered with a predefined frequency to read the file containing the information of the moving elements of the real hall, and to load and display its content
- Preconditions:
  - Input file can be found in the computer's file system and is accessible by the software
  - Input file has been formatted as outlined in README file
  - The timer corresponding to the real hall file is activated from program configurations
  - Input file location and name are those specified by program configurations at the "Real hall file" section
  - Software initialization has completed
- Post-conditions:
  - The data coming from the hall are correctly read by the software and visualized

in the relative tables of the "Moving element data" tab

- [3D hall data previously loaded and automatic discrepancy highlighting turned on] Differences between values from the real hall and the 3D hall relative to the same movable elements are highlighted using the same color for the two cells of the tables in the "Moving element data" tab

- Basic flow:
  1. Timer gets triggered
  2. System searches for real hall data file in the location specified by program configurations
  3. System finds input file and calls Hallbridger API
  4. Hallbridger API parses the file and responds with JSON form
  5. System reads JSON data and fills out Hallbridger inner variables
  6. System displays the loaded data in the corresponding tables of "Moving element data" tab
  7. [3D hall data previously loaded and automatic discrepancy highlighting turned on] System highlights discrepancies with corresponding data from the 3D model in the "Moving element data" tab
- Exception flow: file not found
  1. Timer gets triggered
  2. System searches for real hall data file in the location specified by configuration file
  3. System doesn't find input file and terminates use case without data loaded
- Exception flow: file extension not supported
  1. Timer gets triggered
  2. System searches for real hall data file in the location specified by configuration file
  3. System finds input file with not supported extension
  4. System shows message "File extension not supported"
- Exception flow: error while reading file

1. Timer gets triggered
2. System searches for real hall data file in the location specified by configuration file
3. System finds input file and calls Hallbridger API
4. Hallbridger API raises error while parsing the file and passes error to system
5. System shows message "Error while reading hall data" with error details

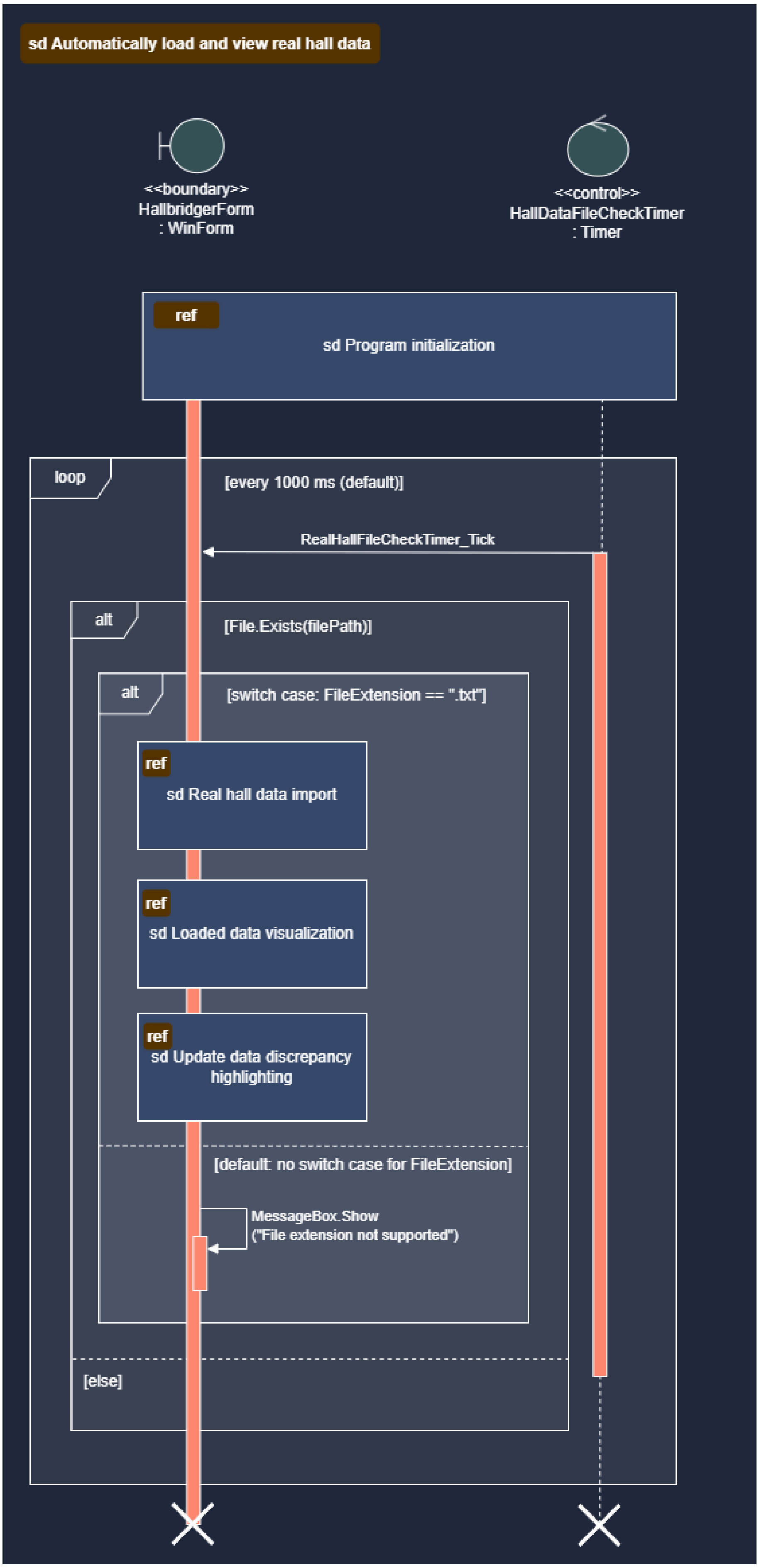


Figure A.1: Sequence diagram for use case "Automatically load and view real hall data".

A.1.2. Automatically load and view 3D hall

- Description: a timer gets triggered with a predefined frequency to read the file containing the 3D model of the hall, and to load and display the data about the position of moving elements contained in it; in addition, the user can view and explore the 3D model in a dedicated viewer, too
- Preconditions:
  - Input file can be found in the computer's file system and is accessible by the software
  - Input file has been formatted as outlined in README file
  - The timer corresponding to the 3D hall file is activated from program configurations
  - Input file location and name are those specified by program configurations at the "3D hall file" section
  - The automatic operation for the timer corresponding to the 3D hall file is set to "Load" in program configurations
  - Software initialization has completed
- Post-conditions:
  - The data about moving element positions contained in the 3D model are correctly read by the software and visualized in the relative tables of the "Moving element data" tab
  - The 3D model is visualized beside a table with reverberation times in the "Acoustic data" tab
  - [Real hall data previously loaded and automatic discrepancy highlighting turned on] Differences between values from the real hall and the 3D hall relative to the same movable elements are highlighted using the same color for the two rows of the tables in the "Moving element data" tab
  - [Moving element configuration in 3D hall recognized] Row of global reverberation time table in the "Acoustic data" tab relative to the 3D hall configuration is bordered in red and the first open pivoting panel array belonging to the current configuration is selected in the 3D model
- Basic flow:

1. Timer gets triggered
2. System searches for 3D hall file in the location specified by program configurations
3. System finds input file
4. System reads file and fills out inner variables
5. System displays the loaded data in the corresponding tables of "Moving element data" tab
6. [Real hall data previously loaded and automatic discrepancy highlighting turned on] System highlights discrepancies with corresponding data from the real hall in the "Moving element data" tab
7. System displays the 3D model in the "Acoustic data" tab
8. [Moving element configuration in 3D hall recognized] System borders in red the row of global RT table in "Acoustic data" tab relative to the current configuration of the 3D hall and selects in the 3D model the first open pivoting panel array belonging to the current configuration

- Exception flow: file not found
  1. Timer gets triggered
  2. System searches for 3D hall file in the location specified by configuration file
  3. System doesn't find input file and terminates use case without data loaded
- Exception flow: file extension not supported
  1. Timer gets triggered
  2. System searches for 3D hall file in the location specified by configuration file
  3. System finds input file with not supported extension
  4. System shows message "File extension not supported"
- Exception flow: error while reading file
  1. Timer gets triggered
  2. System searches for 3D hall file in the location specified by configuration file
  3. System finds input file

4. System shows message "Error while reading 3D model" with error details

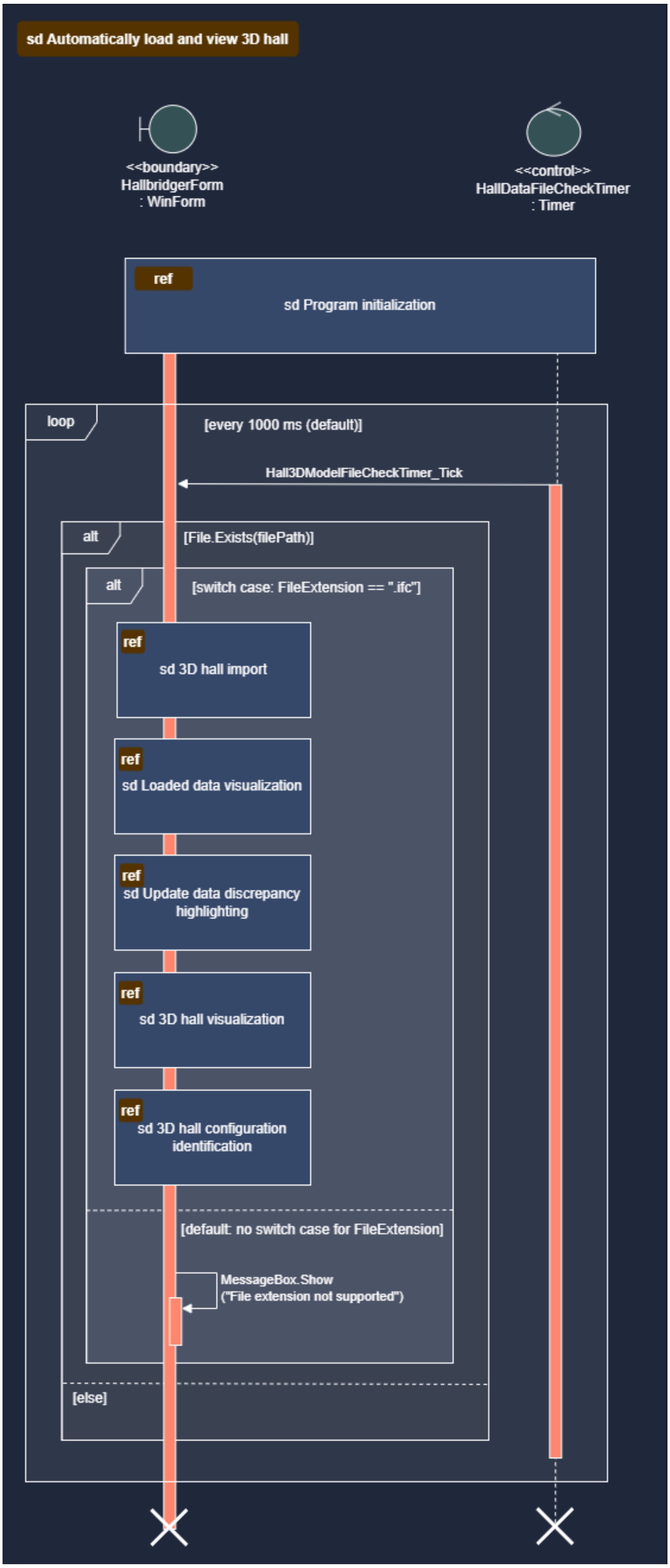


Figure A.2: Sequence diagram for use case "Automatically load and view 3D hall".

A.1.3. Automatically update and view 3D hall

- Description: a timer gets triggered with a predefined frequency to modify the file containing the 3D model of the hall using the previously loaded data coming from the real hall (if the 3D hall file hasn't been loaded yet, the system first loads the file); the software then reloads the file in the interface, displaying both the updated data and 3D model
- Preconditions:
  - Input file can be found in the computer's file system and is accessible by the software
  - Input file has been formatted as outlined in README file
  - The timer corresponding to the 3D hall file is activated from program configurations
  - Input file location and name are those specified by program configurations at the "3D hall file" section
  - The automatic operation for the timer corresponding to the 3D hall file is set to "Update" in program configurations
  - At least some data coming from the real hall have been loaded and are shown in the corresponding tables of "Moving element data" tab
  - Software initialization has completed
- Post-conditions:
  - The "Moving element data" tab shows the updated data relative to moving element positions in the 3D hall
  - The "Acoustic data" tab shows the updated 3D model of the hall
  - [Automatic discrepancy highlighting turned on] Rows of "Moving element data" grids corresponding to items present in both the real and the 3D hall show the same value, so their background color is white
  - [Updated moving element configuration in 3D hall recognized] Row of global reverberation time table in the "Acoustic data" tab relative to the updated 3D hall configuration is bordered in red and the first open pivoting panel array belonging to the current configuration is selected in the 3D model
- Basic flow:

1. Timer gets triggered
2. [No 3D hall previously loaded]
    (a) System searches for 3D hall file in the location specified by program configurations
    (b) System finds input file
    (c) System reads file and fills out inner variables
3. System edits the loaded 3D hall file substituting values of movable element positions with corresponding ones previously loaded from real hall
4. System reloads the 3D hall file and updates inner variables
5. System displays the loaded data in the corresponding tables of "Moving element data" tab
6. [Automatic discrepancy highlighting turned on] System highlights remaining discrepancies between real hall and 3D hall (e.g., for element not loaded for one of the two halls) in the "Moving element data" tab
7. System displays the updated 3D hall in the "Acoustic data" tab
8. [Moving element configuration in 3D hall recognized] System borders in red the row of global RT table in "Acoustic data" tab relative to the current configuration of the 3D hall and selects in the 3D model the first open pivoting panel array belonging to the current configuration

- Exception flow: no real hall data loaded
    1. Timer gets triggered with no real hall data previously loaded
    2. System shows message "No real hall data loaded"
- Exception flow: file not found
    1. Timer gets triggered
    2. System searches for 3D hall file in the location specified by program configurations
    3. System doesn't find input file and terminates use case without data update
- Exception flow: file extension not supported
    1. Timer gets triggered with no 3D hall file previously loaded

2. System searches for 3D hall file in the location specified by program configurations
3. System finds input file with not supported extension
4. System shows message "File extension not supported"

- Exception flow: error while editing file
  1. Timer gets triggered
  2. System searches for 3D hall file in the location specified by program configurations
  3. System finds input file
  4. System tries to edit file and runs into exception
  5. System shows message "Error while modifying 3D model" with error details

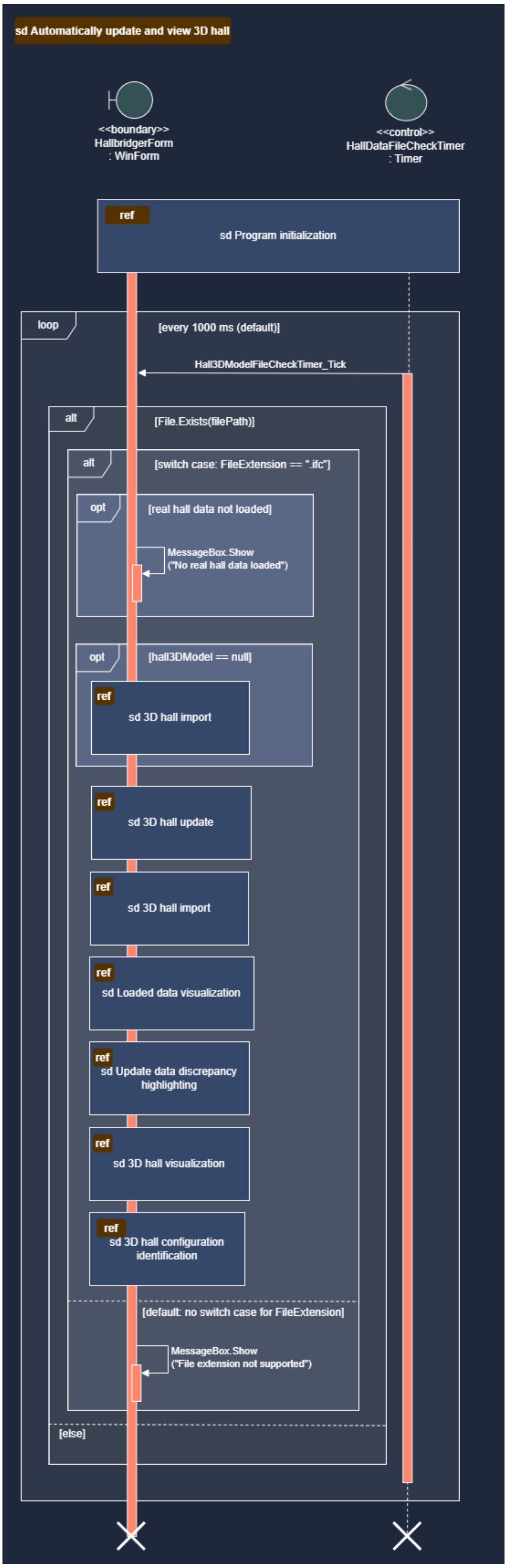


Figure A.3: Sequence diagram for use case "Automatically update and view 3D hall".

A.1.4. Export 3D hall data

- Description: the user chooses a file extension and name and an output directory, and the program creates an export file with those extension and name in the chosen location containing data about the position of moving elements in the previously loaded 3D model of the hall
- Preconditions:
  - A 3D hall has been loaded, so its data are shown in the corresponding tables of "Moving element data" tab and its model appears in the "Acoustic data" tab
  - Desired output file location is accessible by the computer's file system
  - Software initialization has completed
- Post-conditions:
  - An output file with the user-defined name and extension is located in the chosen directory
  - The output file contains the data about moving element positions displayed in the tables of "Moving element data" tab relative to the 3D hall
  - The output file is formatted as outlined in the README file and is ready to be read by the theater's stage control software
- Basic flow:
  1. Operator navigates to "Moving element data" tab and presses "Export 3D hall data" button
  2. System shows window with file explorer
  3. Operator chooses output file name and extension, navigates the file system and selects directory to save output file in
  4. System reads inner variables and fills out a JSON form
  5. System calls Hallbridger API passing JSON form
  6. Hallbridger API reads JSON data and creates output file with the specified name and extension
  7. System saves output file in the specifies directory

- Exception flow: no 3D hall loaded
  1. Operator navigates to "Moving element data" tab and presses "Export 3D hall data" button with no 3D hall data previously loaded
  2. System shows message "No 3D hall data loaded"
- Exception flow: error while creating file
  1. Operator navigates to "Moving element data" tab and presses "Export 3D hall data" button
  2. System shows window with file explorer
  3. Operator chooses output file name and extension, navigates the file system and selects directory to save output file in
  4. System reads inner variables and fills out a JSON form
  5. System calls Hallbridger API passing JSON form
  6. Hallbridger API raises error while creating output file and passes error to system
  7. System shows message "Error while exporting 3D model data to file" with error details

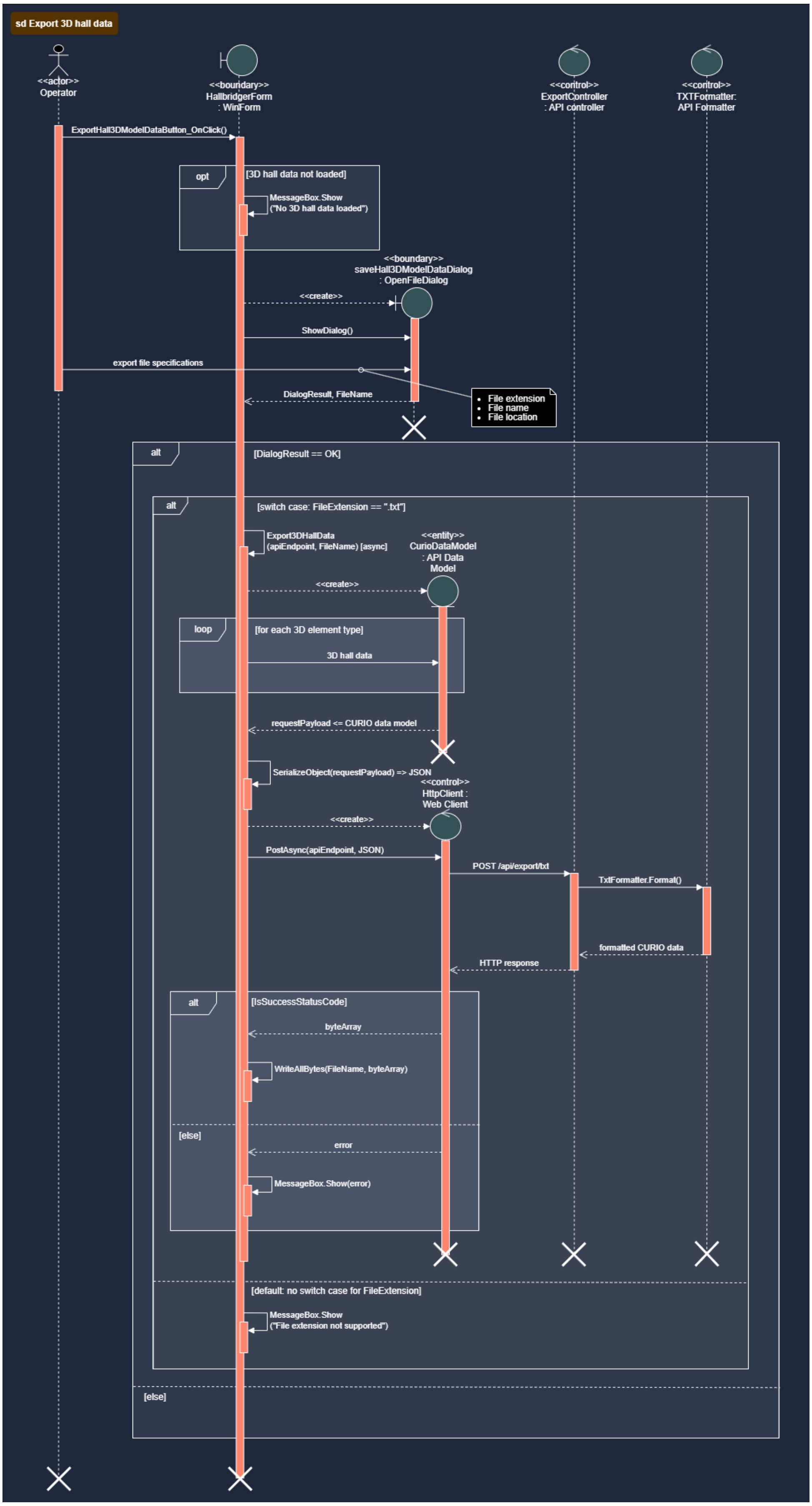


Figure A.4: Sequence diagram for use case "Export 3D hall data".

A.1.5. Show/hide data discrepancy highlighting

- Description: the user clicks on "Highlight data discrepancies" checkbox and the system shows (checked) or hides (unchecked) the highlighting of differences between data about the same moving element in real and 3D hall, by means of coloring table rows containing corresponding data the same way
- Preconditions:
  - Software initialization has completed
  - At least some data has been loaded for both real and 3D hall
- Post-conditions:
  - [checked] Different values for the same element between real and 3D hall are colored the same
  - [unchecked] The grids' background color is the default one in "Moving element data" tab
- Basic flow:
  1. Operator navigates to "Moving element data" tab and clicks on the "Highlight data discrepancies" checkbox
  2. [checked] System applies the same color to the background of grid row couples in "Moving element data" tab relative to the same moving element between real and 3D hall and containing different values
  3. [unchecked] System applies the default color to every row of every grid of "Moving element data" tab

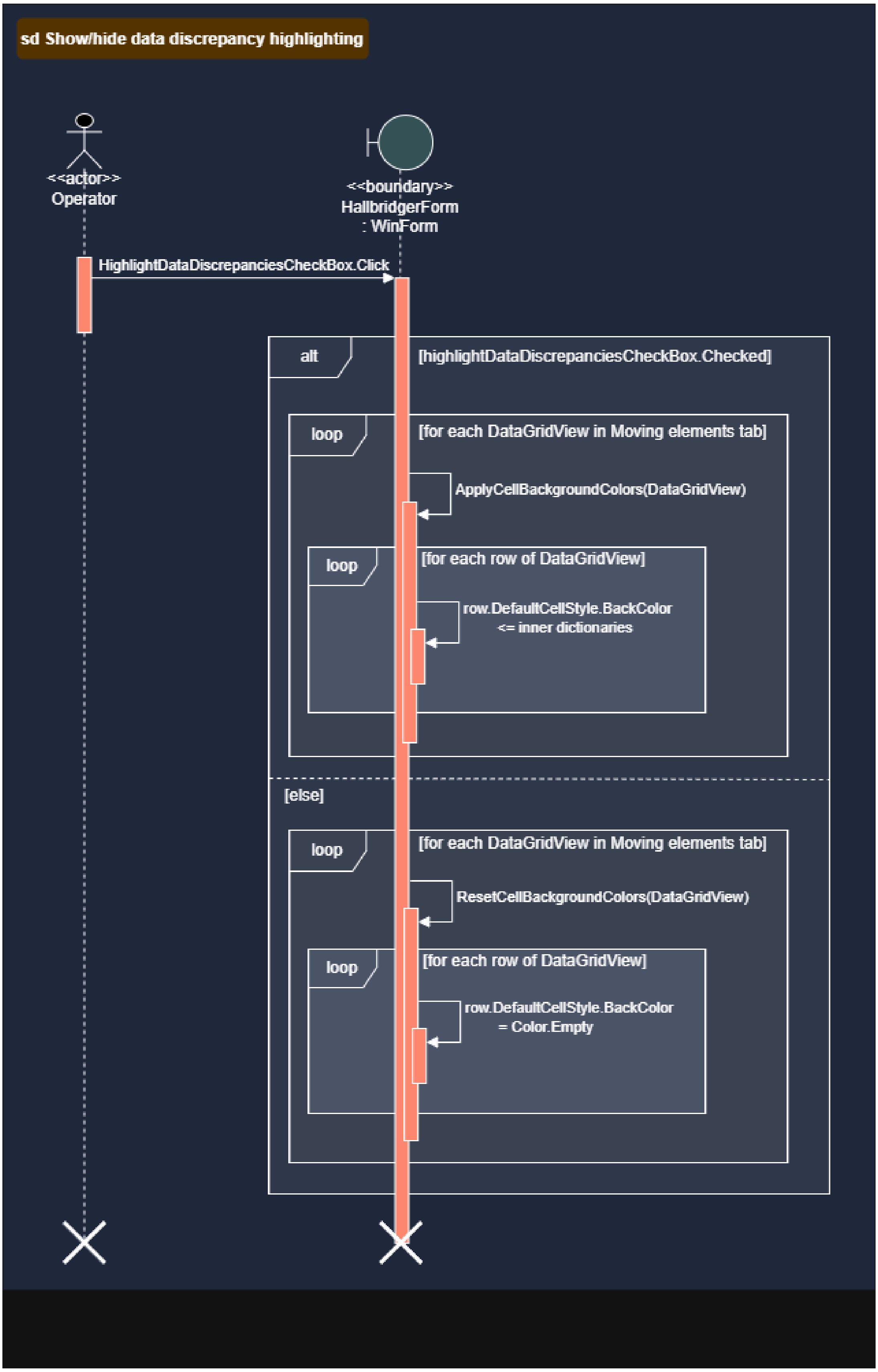


Figure A.5: Sequence diagram for use case "Show/hide data discrepancy highlighting".

A.1.6. Enable/disable automatic data discrepancy highlighting

- Description: the user clicks on the "Automatic discrepancy highlighting" option in the "Options" menu and the program activates (checked) or deactivates (unchecked) the automatic highlighting of differences between real and 3D hall data when a new input file is loaded
- Preconditions:
  - Software initialization has completed
- Post-conditions:
  - [checked] Automatic highlighting is enabled, so the next time an input file is loaded, if data from both real and 3D hall are present, discrepancies between corresponding data are going to be highlighted with the same color
  - [unchecked] Automatic highlighting is disabled, so the next time an input file is loaded, if data from both real and 3D hall are present, discrepancies between corresponding data are not going to be shown and cell background color is going to be the default one
- Basic flow:
  1. Operator navigates to "Options" menu on the menu bar
  2. Operator clicks on "Automatic discrepancy highlighting" menu entry
  3. [checked] System activates the automatic data discrepancy highlighting
  4. [unchecked] System deactivates the automatic data discrepancy highlighting

A.1.7. View acoustic data

- Description: the user navigates to the "Acoustic data" tab of the interface and views a table with global reverberation times, divided in octave bands, relative to various open/close configurations of the pivoting panels of the hall
- Preconditions:
  - Software initialization has completed
- Post-conditions:
  - The user sees the "Acoustic data" tab on screen

- Basic flow:
    1. Operator navigates to "Acoustic data" tab
    2. Operator views the the hall's global reverberation times

A.1.8. Reposition 3D model

- Description: the user clicks on the "Reposition" button in the "Acoustic data" tab and the camera in the 3D viewer moves back to the initial position
- Preconditions:
    - Software initialization has completed
    - A 3D model has been loaded and is visualized in the 3D viewer
- Post-conditions:
    - The view of the 3D model is set back to when it was just loaded
- Basic flow:
    1. Operator navigates to "Acoustic data" tab and presses "Reposition" button
    2. System sets the point of view in the 3D viewer back to the starting position

A.1.9. Enable/disable high performance mode

- Description: the user clicks on the "High performance mode" checkbox in the "Acoustic data" tab and the 3D viewer enables (checked) or disables (unchecked) a lower-resolution rendering for the 3D model to accommodate slower video processing capabilities and use less graphics memory
- Preconditions:
    - Software initialization has completed
- Post-conditions:
    - [checked] The 3D model hosted by the viewer in the "Acoustics data" tab operates in "high performance mode", so the model it shows is rendered with less definition than in normal mode
    - [unchecked] The 3D model operates normally, showing the 3D model with highest possible definition and exploiting all the graphics resources at disposal

- Basic flow:
  1. Operator navigates to "Acoustic data" tab and clicks on the "High performance mode" checkbox
  2. [checked] System switches 3D viewer to "High performance mode", visualizing the model with a lower graphic resolution
  3. [unchecked] System switches 3D viewer to ordinary mode, visualizing the model with its original resolution

A.1.10. View and edit 3D element properties

- Description: the user selects an element of the 3D model in the viewer and a popup window appears listing the main properties of the element; if it is a moving element recognized by Hallbridger the user can also edit the value of the position (if it is a stagecraft equipment piece) or aperture angle (if it is a pivoting panel array), modifying the 3D model directly. Multiple selection is possible, holding CTRL while clicking on the elements. Deselecting an element (holding CTRL) makes its property window disappear.
- Preconditions:
  - Software initialization has completed
  - A 3D hall has been loaded, so its data are shown in the corresponding tables of "Moving element data" tab and its model appears in the "Acoustic data" tab
- Post-conditions:
  - A small popup window shows up containing the 3D element main properties
  - [Moving element recognized] The value specified by the user for the position/aperture property is saved to the 3D model
- Basic flow:
  1. Operator navigates to "Acoustic data" tab and clicks on one (or more, holding CTRL) element(s) of the 3D model that appears in the viewer
  2. System retrieves information about the selected 3D element(s) and opens a small window for each of them, containing the retrieved information
  3. [Moving element recognized]

   (a) Operator edits the position/aperture angle value for one of the selected elements choosing one from the dropdown list, using the up/down buttons or directly typing in the desired value, and confirms pressing Enter or the "Apply" button
   (b) System checks the new value for validation
   (c) System updates inner dictionaries, 3D model file and views
   (d) System shows a success message
4. Operator deselects a selected 3D element by holding CTRL pressed and clicking on it
5. System closes the relative popup window

- Exception flow: invalid inserted value
   1. Operator navigates to "Acoustic data" tab and clicks on one (or more, holding CTRL) element(s) of the 3D model that appears in the viewer
   2. System retrieves information about the selected 3D element(s) and opens a small window for each of them, containing the retrieved information
   3. Operator types in a not valid value for one of the selected recognized moving elements' position/aperture angle value, or leaves the text field empty, and presses Enter or the "Apply" button
   4. System checks the new value for validation
   5. System shows an "invalid value" tooltip for some seconds, with the indication of allowed inputs
   6. System sets the displayed value back to the last valid value
- Exception flow: inserted value out of bounds
   1. Operator navigates to "Acoustic data" tab and clicks on one (or more, holding CTRL) element(s) of the 3D model that appears in the viewer
   2. System retrieves information about the selected 3D element(s) and opens a small window for each of them, containing the retrieved information
   3. Operator types in a value greater than the maximum allowed or smaller than minimum for one of the selected recognized moving elements' position/aperture angle value, and confirms pressing Enter or the "Apply" button

4. System checks the new value for validation
5. System shows a "value out of bounds" tooltip for some seconds, with the indication of min-max allowed interval
6. System sets the displayed value back to the last valid value

- Exception flow: new value equaling old value
  1. Operator navigates to "Acoustic data" tab and clicks on one (or more, holding CTRL) element(s) of the 3D model that appears in the viewer
  2. System retrieves information about the selected 3D element(s) and opens a small window for each of them, containing the retrieved information
  3. Operator presses Enter or the "Apply" button for one of the selected recognized moving elements while its position/aperture angle value displayed is the same as the last saved one
  4. System checks the new value for validation
  5. System does not show any message and does not modify any value
- Exception flow: error while editing file
  1. Operator navigates to "Acoustic data" tab and clicks on one (or more, holding CTRL) element(s) of the 3D model that appears in the viewer
  2. System retrieves information about the selected 3D element(s) and opens a small window for each of them, containing the retrieved information
  3. Operator edits the position/aperture angle value for one of the selected elements choosing one from the dropdown list, using the up/down buttons or directly typing in the desired value, and confirms pressing Enter or the "Apply" button
  4. System checks the new value for validation
  5. System tries to update value and runs into exception
  6. System shows message "Error while modifying value" with error details

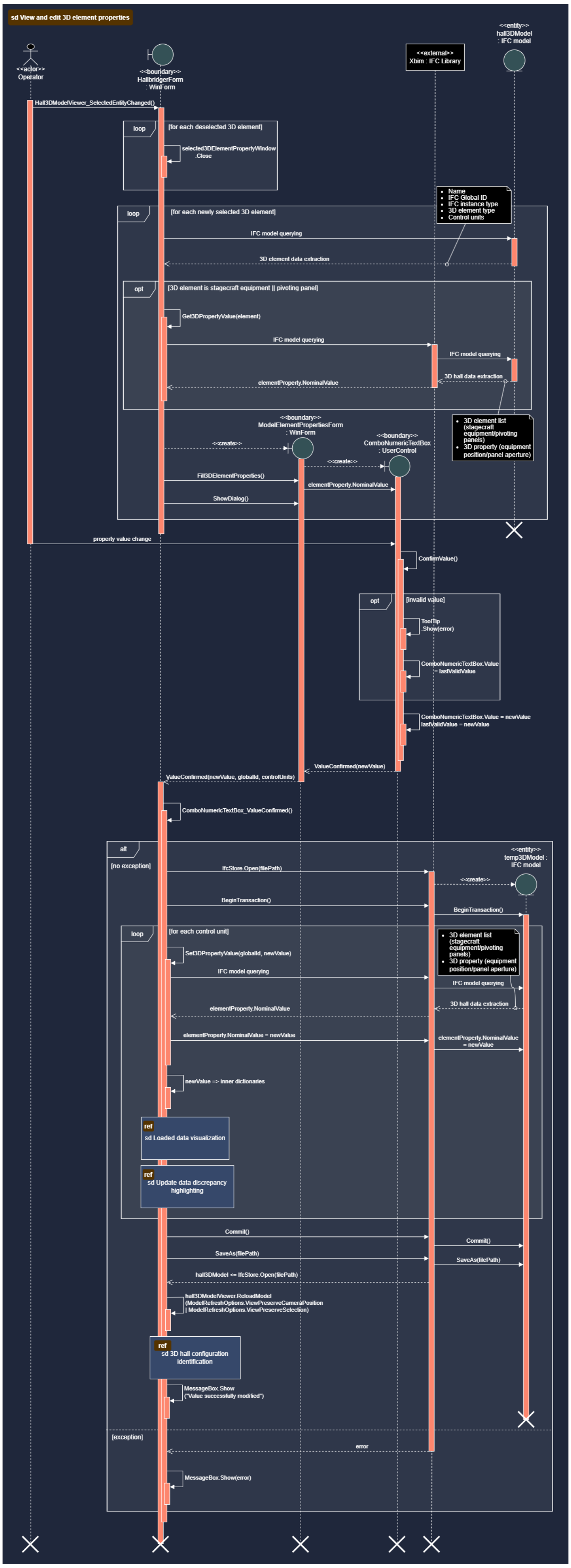


Figure A.6: Sequence diagram for use case "View and edit 3D element properties".

A.1.11. Edit software preferences

- Description: the user opens the configuration window, from which they can view and edit the program's current preferences about automatic input file reading, loading and updating, and the software is re-initialized to apply them after confirmation
- Preconditions:
  - Software initialization has completed
- Post-conditions:
  - The new software configurations are stored in the INI file
  - The software behaves according to the new user preferences
- Basic flow:
  - Operator navigates to "Options" menu on the menu bar
  - Operator clicks on "Configurations" menu entry
  - System retrieves software settings, opens configuration window and displays them
  - Operator edits preferences and confirms them clicking OK button
  - System checks the new values for validation
  - System updates configuration file
  - System re-initializes program applying changes
- Exception flow: invalid inserted values
  - Operator navigates to "Options" menu on the menu bar
  - Operator clicks on "Configurations" menu entry
  - System retrieves software settings, opens configuration window and displays them
  - Operator edits preferences using invalid values and confirms them clicking OK button
  - System checks the new values for validation
  - System shows an "invalid value" message, with the indication of allowed inputs

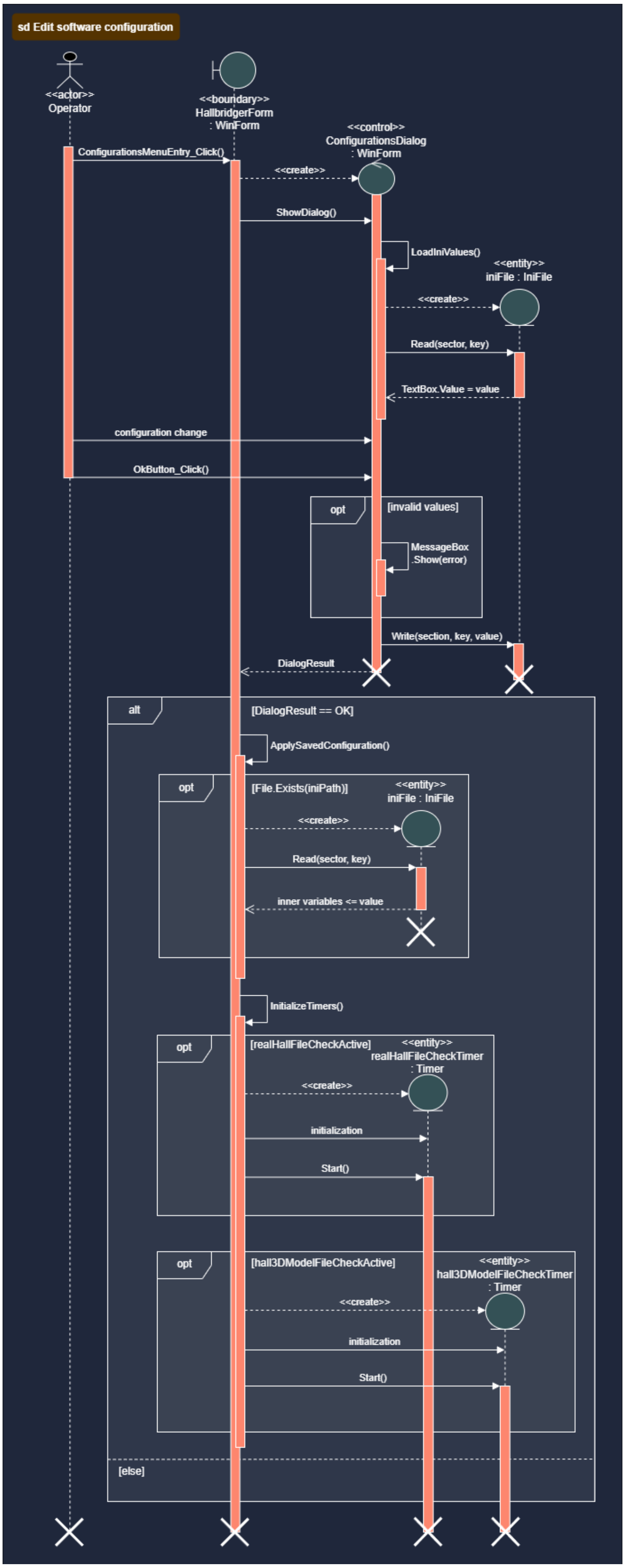


Figure A.7: Sequence diagram for use case "Edit software configuration".

A.2. Auxiliary sequence diagrams

The referenced sequence diagrams ("ref" blocks in those of 3.4 and A.1) are included in this section.

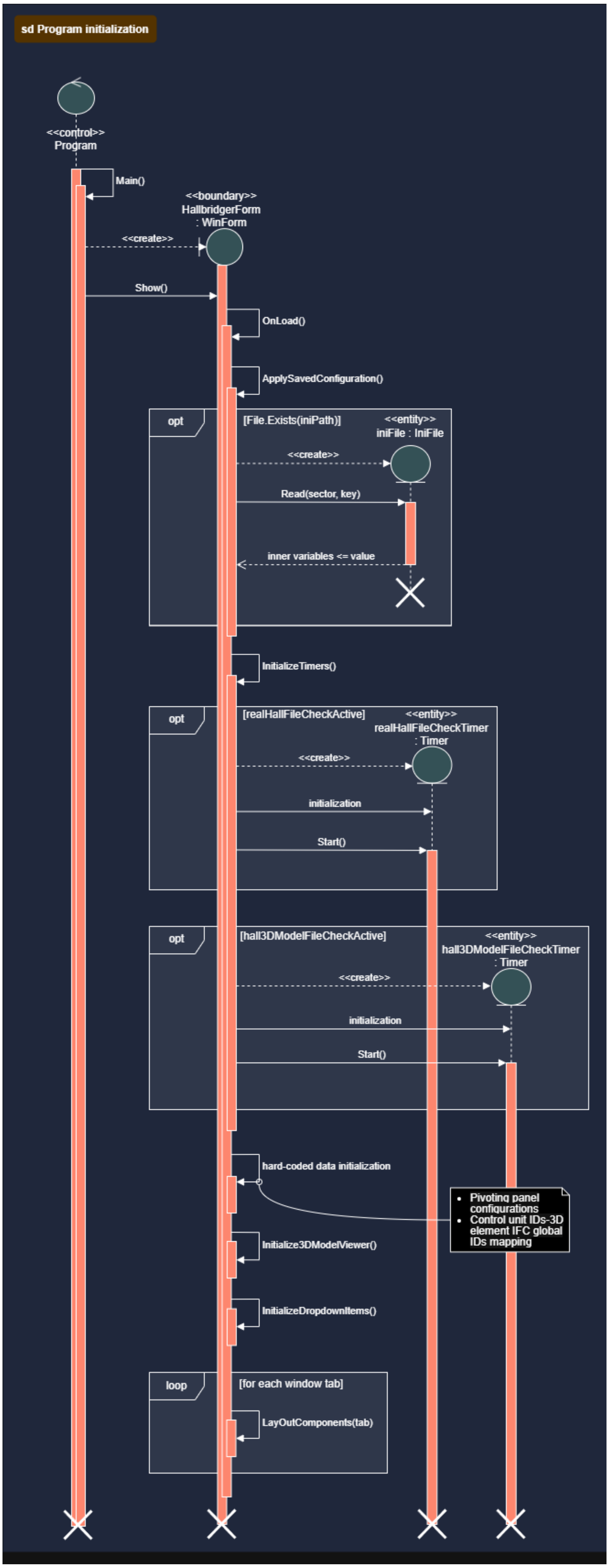


Figure A.8: Sequence diagram for action "Program initialization".

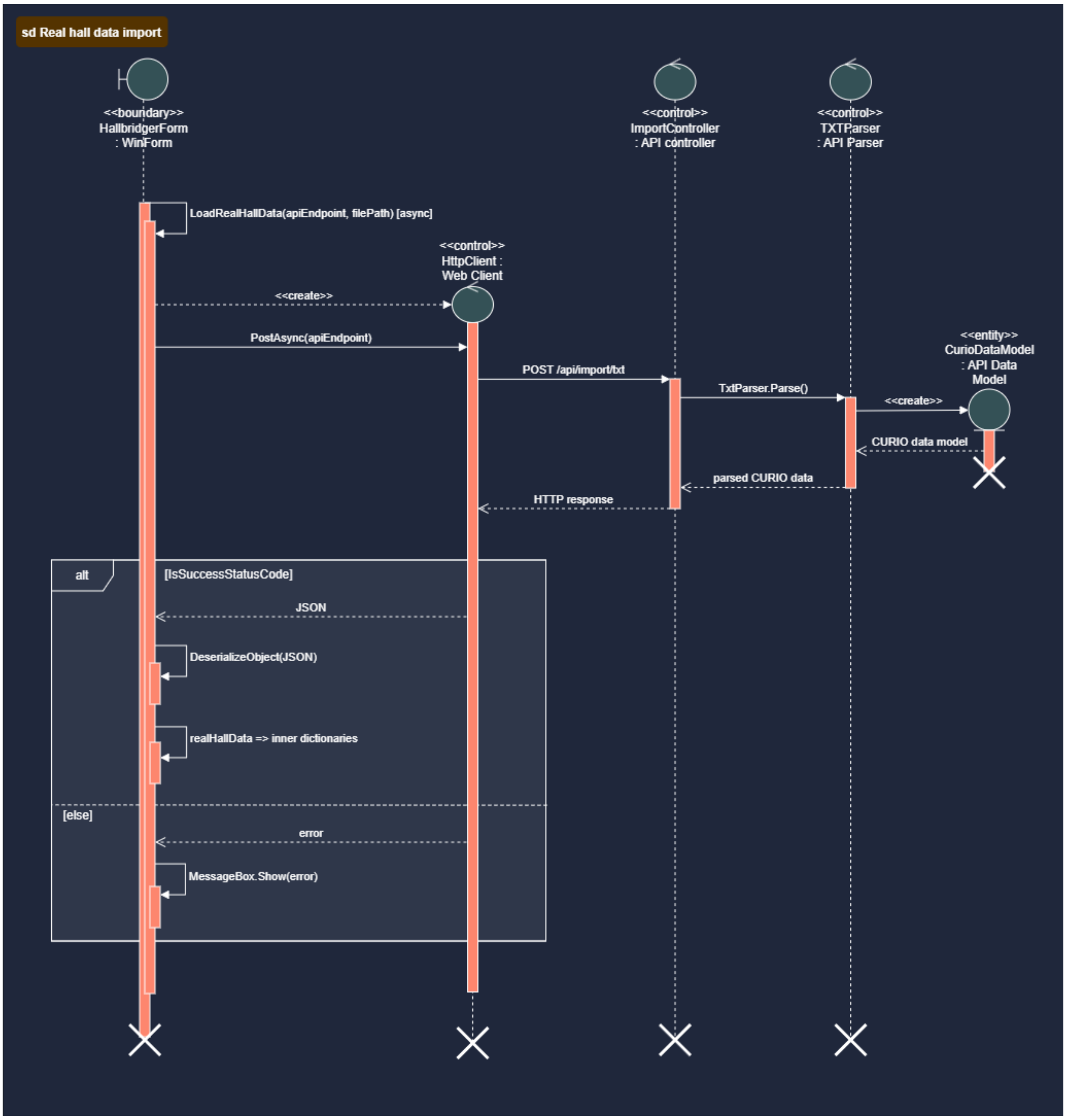


Figure A.9: Sequence diagram for action "Real hall data import".

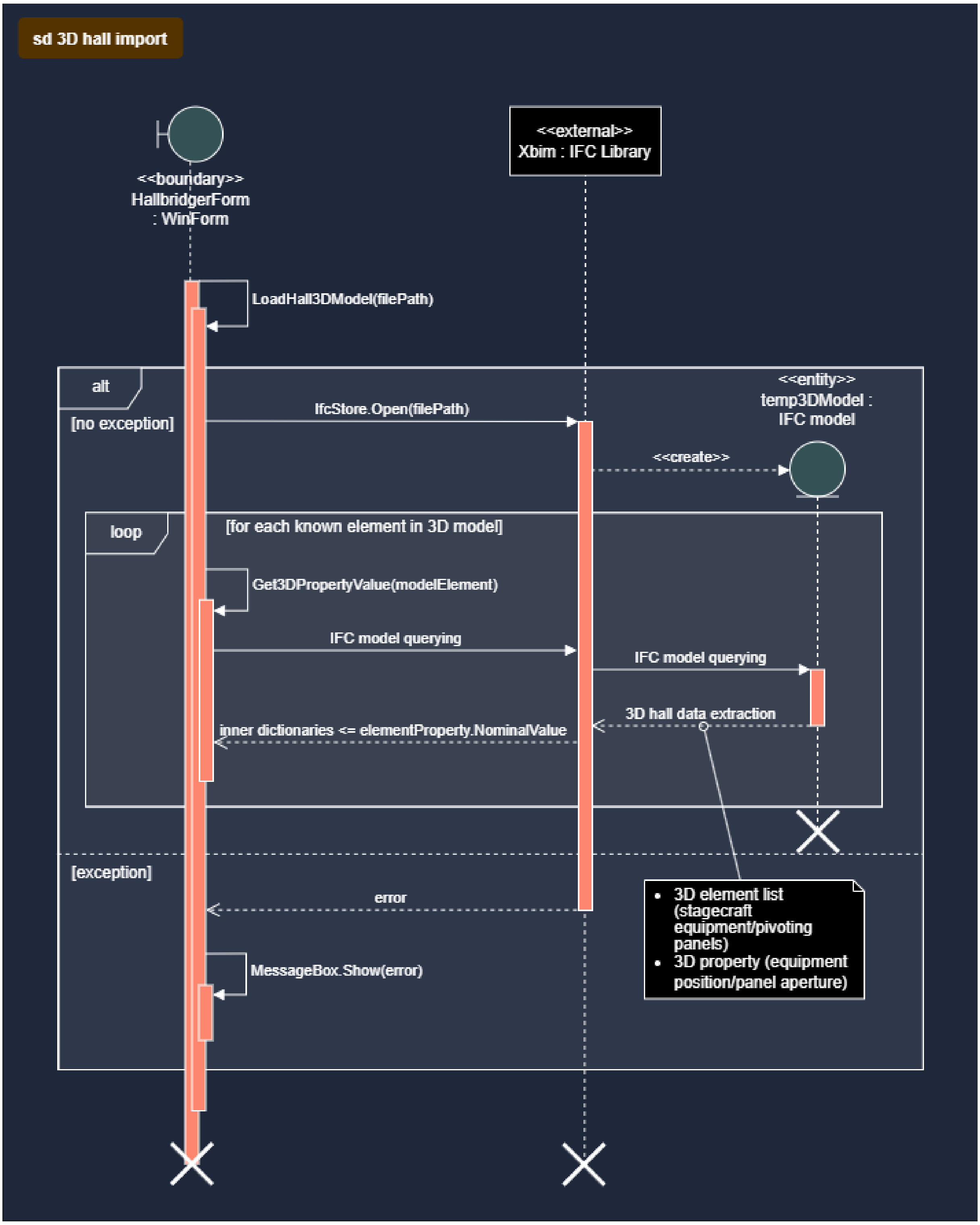


Figure A.10: Sequence diagram for action "3D hall import".

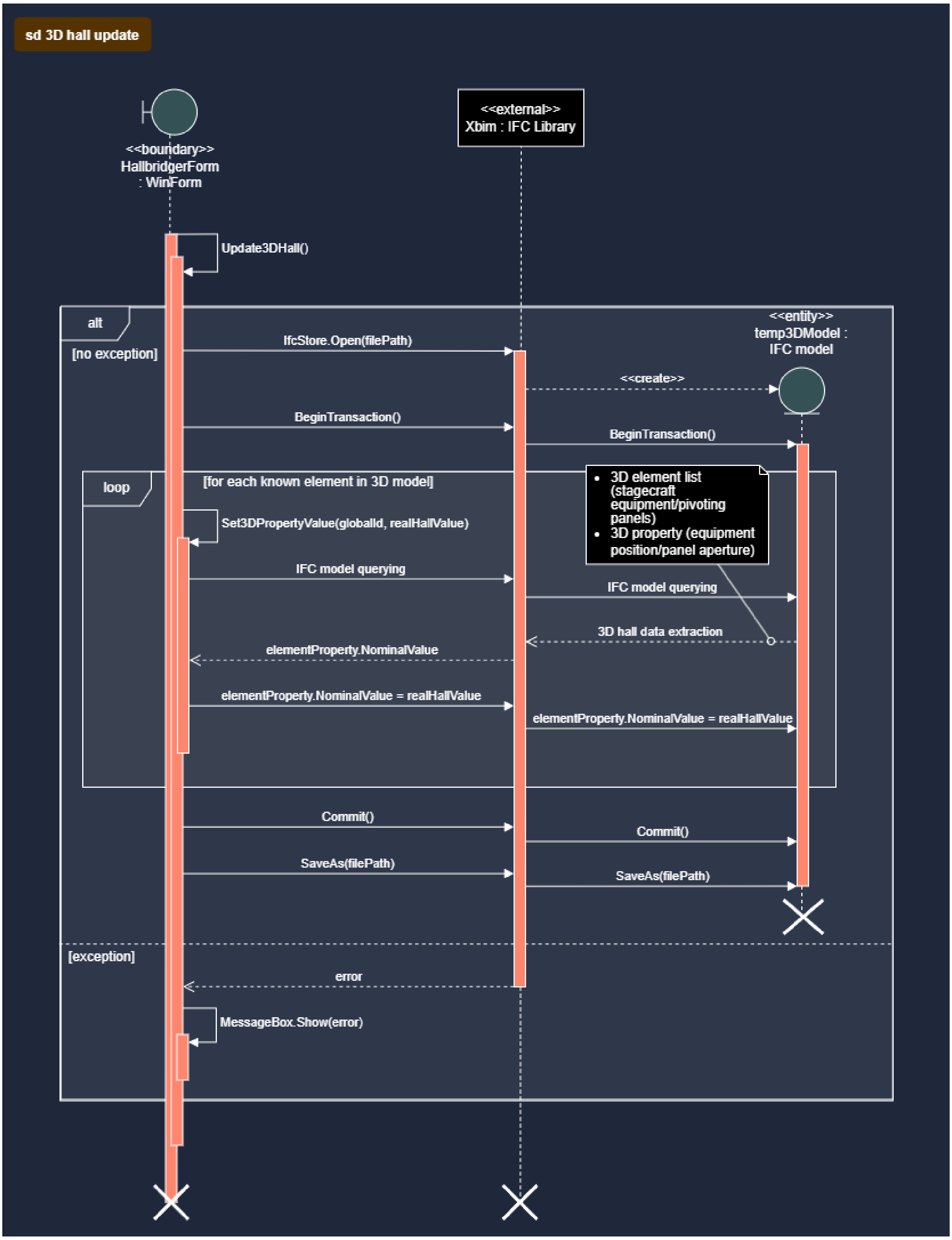


Figure A.11: Sequence diagram for action "3D hall update".

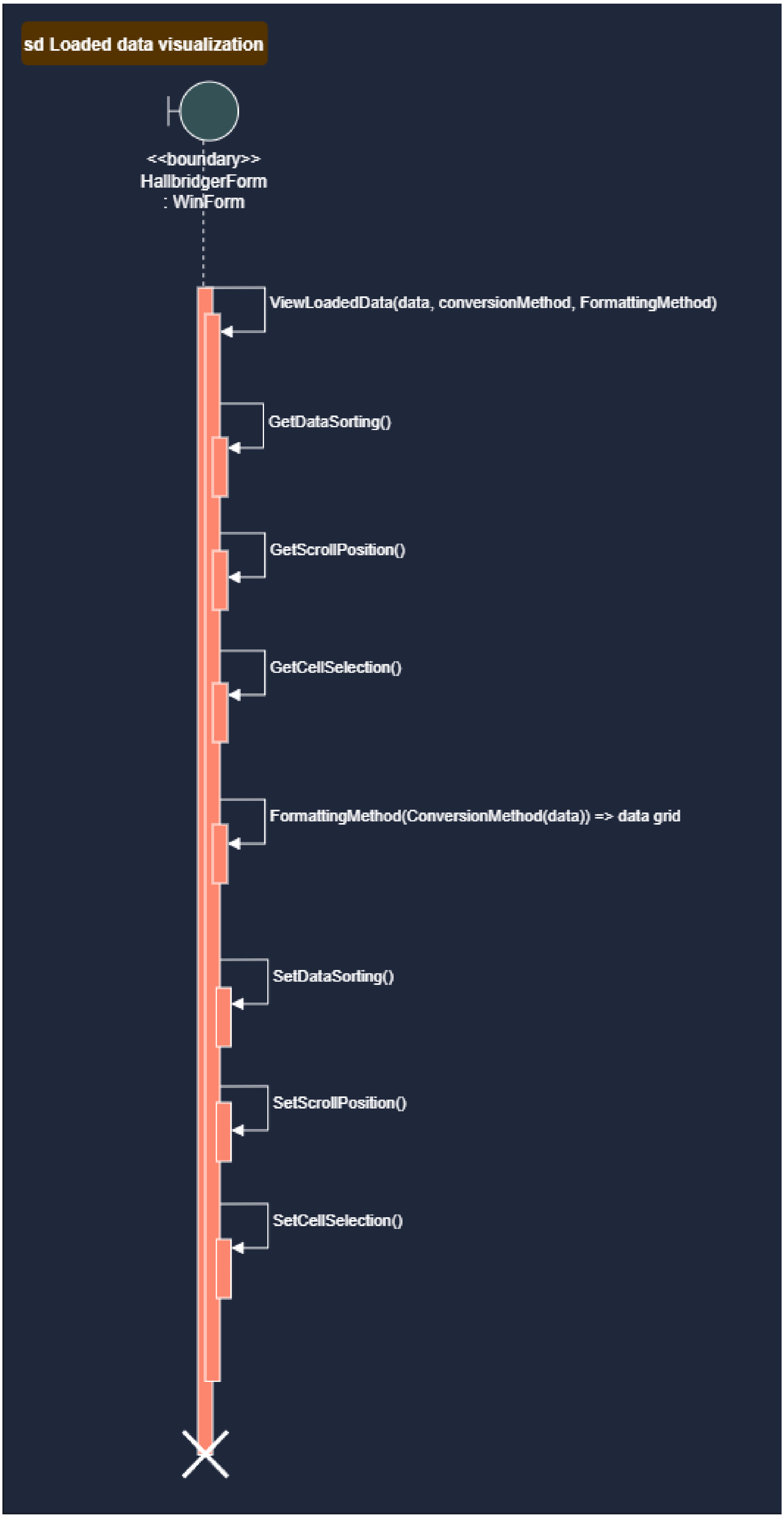


Figure A.12: Sequence diagram for action "Loaded data visualization".

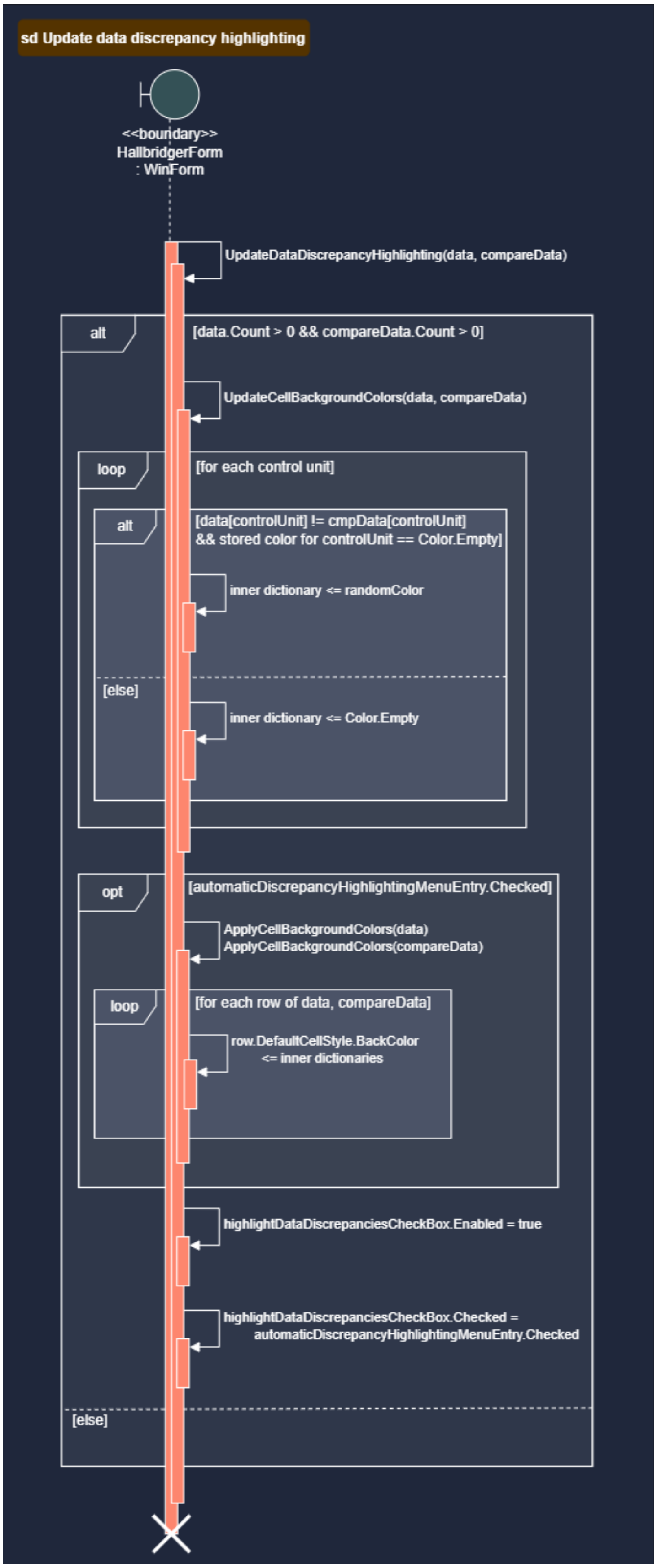


Figure A.13: Sequence diagram for action "Update data discrepancy highlighting".

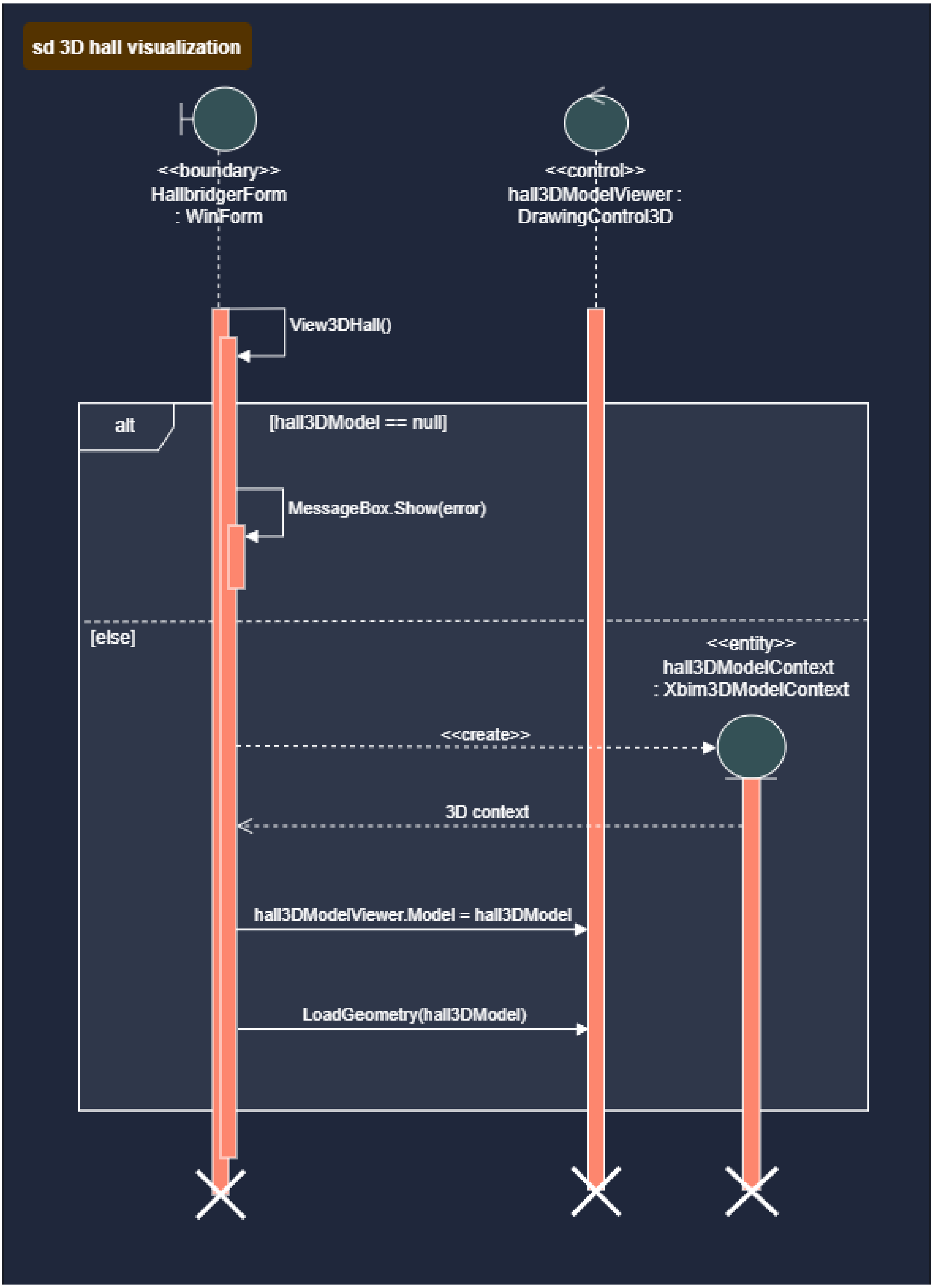


Figure A.14: Sequence diagram for action "3D hall visualization".

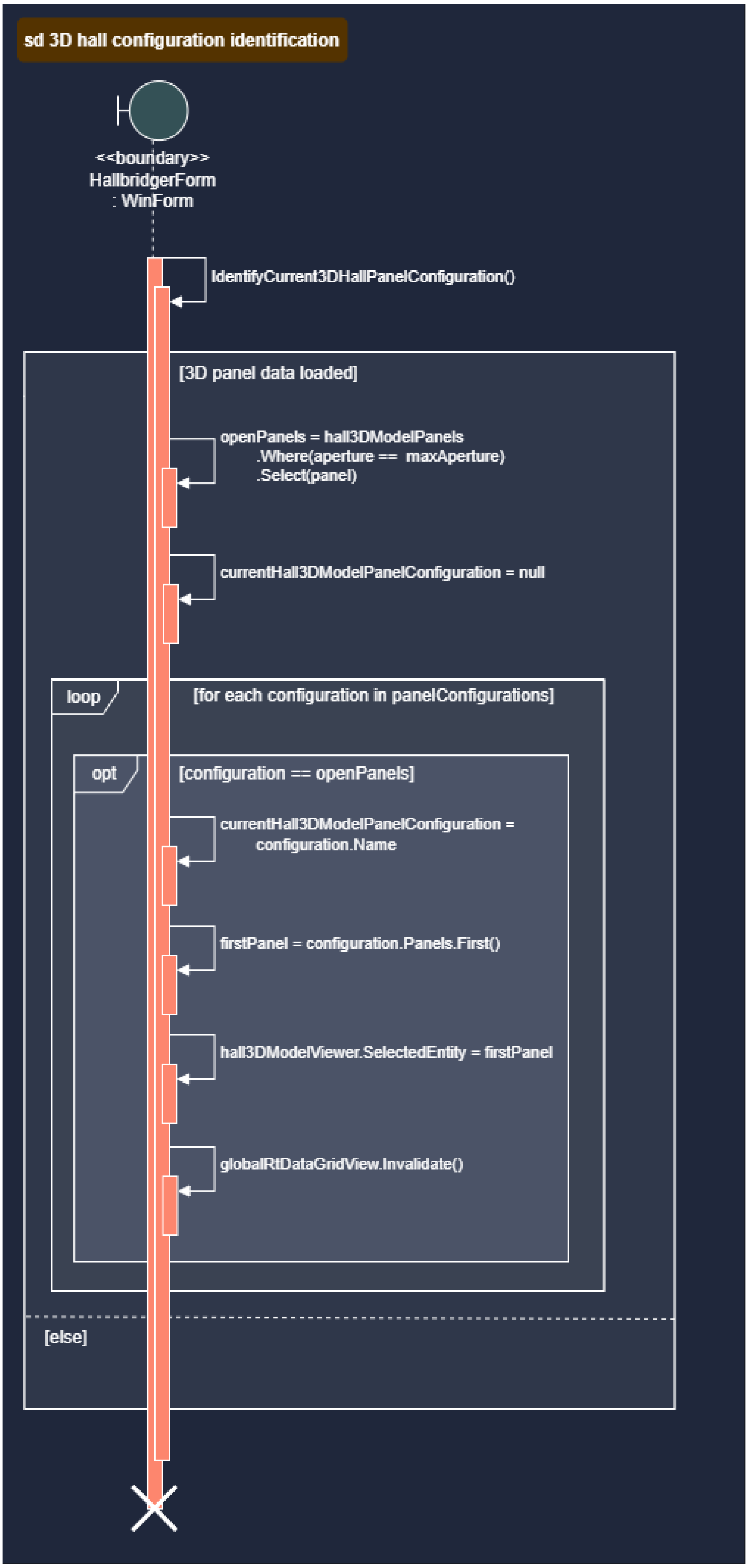


Figure A.15: Sequence diagram for action "3D hall configuration identification".

List of Figures

List of Tables

List of Acronyms

Acronym	Description
FM	Facility Management
BIM	Building Information Modeling
DT	Digital Twin
IoT	Internet of Things
VR	Virtual Reality
RT	Reverberation Time
GUI	Graphical User Interface
API	Application Program Interface

Acknowledgements

Se sono riuscito a terminare questo percorso è grazie all'amore di chi mi è sempre stato accanto, tanto grande da insegnarmi ad amare me stesso. Grazie alla mia famiglia, i miei genitori, agli amici di sempre e a quelli fatti a Padova e a Milano, a Irene con cui ho condiviso tanti anni, a Elisabetta per essere stata con me lungo tutta questa tesi, e per esserlo ancora adesso che si è conclusa; Grazie al prof. Avanzini e al dott. Geronazzo, che non ho ringraziato ufficialmente nella tesi triennale, per avermi dato la possibilità dell'esperienza di ricerca accademica.