O_HAI(4)Games

Orchestration of Hybrid Artificial Intelligence Methods for Computer Games

Literature Review

This project was funded by the Croatian Science Foundation

Principal investigator:

Markus Schatten

PUBLISHED BY ARTIFICIAL INTELLIGENCE LABORATORY,
FACULTY OF ORGANIZATION AND INFORMATICS, UNIVERSITY OF ZAGREB

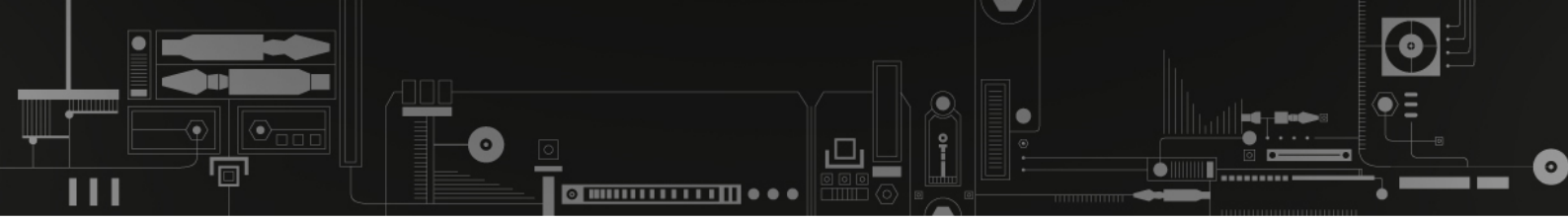
HTTP://AI.FOI.HR/OHAI4GAMES

Some of the results presented in this deliverable have been previously published in [53, 54, 59].

Technical Report No. AIL202002 – First release, July 2020, edit April 2021

Document compiled by: Markus Schatten with inputs from other project team members

This work has been supported in full by the Croatian Science Foundation under the project number IP-2019-04-5824.

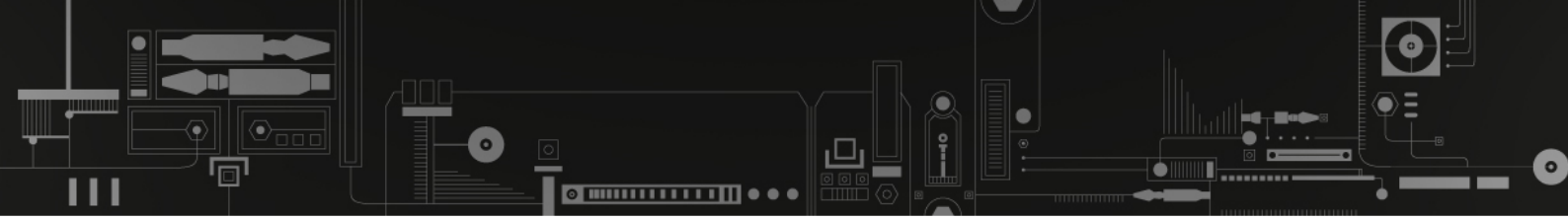


hrzz
Hrvatska zaklada za znanost

foi
SVEUČILIŠTE U ZAGREBU
FAKULTET
ORGANIZACIJE I
INFORMATIKE
VARAŽDIN

lab

Contents



hrzz
Hrvatska zaklada za znanost

foi
SVEUČILIŠTE U ZAGREBU
FAKULTET
ORGANIZACIJE I
INFORMATIKE
VARAŽDIN

lab

1. Project Description

1.1 Abstract

HAI methods, which can be defined as the orchestration of complementary heterogeneous both symbolic and statistical AI methods to acquire more precise results, are omnipresent in contemporary scientific literature. Still, the methodology of developing such systems is in most cases ad-hoc and depends from project to project. Computer games have always been connected to the development of AI. From the earliest chess minmax algorithm by Claude Shannon in 1949 to the more recent AlphaGo in 2015, computer games provide an ideal testing environment for AI methods. Similarly, AI has always been an important part of computer games, which have often been judged by the quality of their AI and praised if they used an innovative approach. Computer games allow us to test AI methods, not only for fun and leisure, but also for numerous other fields of human activity through the fields of serious games and gamification. The project proposes to establish an new framework for the orchestration of hybrid artificial intelligence methods with a special application to computer games. Therefore an ontology of hybrid AI methods as well as a meta-model shall be developed that would allow for creating models (ensembles) of hybrid AI methods. This meta-model would be implemented into a modular distributed orchestration platform which would be further enriched with a number of modules to be tested in four gaming related environments: (1) MMORPG games, (2) gamified learning platform, (3) serious game related to autonomous vehicles and (4) a game for a holographic/volumetric gaming console which would also be developed during the project.

1.2 Introduction

The application of HAI which can be defined as the orchestration of heterogeneous artificial intelligence (AI) methods including both statistical and symbolic approaches in various domains is omnipresent in current scientific literature. It is largely overlapping with the term hybrid intelligence (HI) that has been defined as *"the combination of complementary heterogeneous intelligences (...) to create a socio-technological ensemble that is able to overcome the current limitations of (artificial) intelligence."* [18]. HI lies at the intersection of human, collective and

artificial intelligence, with the intent of taking the best of each.

There have been numerous studies recently addressing issues related to HAI and HI methods in a multitude of application domains including but not limited to land-slide prediction [40], drug testing [11], forecasting crude oil prices [70], prediction of wildfire [29], evaluation of slope stability [32], modeling of hydro-power dam [10], wind energy resource analysis [21], industry 4.0 and production automation [6], airblast prediction [5], heart disease diagnosis [41] and these are just a few references from 2018 until the time of writing this proposal. Most of these and such studies report building HAI systems by combining various AI methods to acquire better and more precise results. However, when it comes to methodology of the actual orchestration of HAI methods the usual approach is ad-hoc and depends from project to project. The lack of methodology in orchestrating HAI shall be addressed in the proposed project.

In a previous project sponsored by the Croatian Science Foundation (Installation Project No. HRZZ-UIP-2013-11-8537 entitled Large-Scale Multi-Agent Modelling of Massively Multi-Player On-Line Playing Games - ModelMMORPG - see [62] for details) a comprehensive methodology for modelling large-scale intelligent distributed systems has been developed that includes a graphical modelling tool and code generator (described in [61] and in more detail in [43]). The implemented toolset allows for modelling complex multi-agent organizations and could be applied to numerous applications domains [57, 58]. Herein, we would like to apply and incorporate this methodology to the development of a HAI orchestration platform.

Computer games have always been connected to the development of AI. From the earliest chess minmax algorithm by Claude Shannon in 1949 to the more recent AlphaGo™ in 2015, computer games provide an ideal testing environment for AI methods. Similarly, AI has always been an important part of computer games. Computer games have often been judged by the quality of their AI and praised if they used an innovative approach like the ghosts in Pacman™ which had individual personality traits (1980), Creatures™ which used neural networks for character development (1996), Black & White™ which used the belief-desire-intention (BDI) model (2000), F.E.A.R.™ which used automated planning algorithms (2005) and many others (see [78, pp. 8–15] for a very detailed overview). Artificial intelligence in games is not only used for non-playing character (NPC) or opponent implementation, but also for various other parts of games [78, pp. 151–203] including but not limited to **generation of content** (graphics including levels and maps, sound, narratives, rules and mechanics or even whole games like the Angelina game-generating system [15]), **player behaviour and experience modeling** [78, pp. 203–259], as well as **bot development and automated game testing** [78, pp. 91–151]. Due to their complex nature and endless possibilities of creative design, computer games present us with an excellent opportunity to study the orchestration of HAI in various scenarios – not only for fun and leisure but also for other domains in form of serious games and/or gamification.

In the previously mentioned ModelMMORPG project, we have already used an open source massively multi-player on-line role-playing game (MMORPG) called The Mana World (TMW) for which we have implemented a high-level interface to test intelligent agents playing the game. Additionally a number of connected game quests have been developed for various scenarios which allowed us to build an automated game testing system [60]. Herein we would like to use this interface to test orchestrated HAI methods, but also develop other testbeds for the planned platform.

Therefore, the main contribution of the proposed project shall be: (1) a comprehensive framework for the orchestration of hybrid artificial intelligence methods for computer games allowing to define models of HAI for various purposes, (2) an open source distributed cloud platform that will allow to implement such models based on existing HAI methods and connect them directly from game development platforms, (3) a set of best practices in developing HAI ensemble models tested in at least four specific testbeds described bellow.

1.3 Team Members

Markus Schatten (Principal investigator)
Head of Artificial Intelligence Laboratory,
Faculty of Organization and Informatics,
University of Zagreb

Jaime Andres Rincon Arango
Grupo de Tecnología Informática e Inteligencia Artificial (GTI-IA),
Departamento de Sistemas Informáticos y Computación,
Universitat Politècnica de València

Bogdan Okreša Đurić
Artificial Intelligence Laboratory,
Faculty of Organization and Informatics,
University of Zagreb

Carlos Carrascosa
Grupo de Tecnología Informática e Inteligencia Artificial (GTI-IA),
Departamento de Sistemas Informáticos y Computación,
Universitat Politècnica de València

Damir Horvat
Department of Quantitative Methods
Faculty of Organization and Informatics,
University of Zagreb

Vicente Julian
Grupo de Tecnología Informática e Inteligencia Artificial (GTI-IA),
Departamento de Sistemas Informáticos y Computación,
Universitat Politècnica de València

Tomislav Peharda
Artificial Intelligence Laboratory,
Faculty of Organization and Informatics,
University of Zagreb

Glenn Smith
College of Education,
University of South Florida

Igor Tomičić
Center for Forensics, Biometrics and Privacy
Faculty of Organization and Informatics,
University of Zagreb

Neven Vrček
Department of Information Systems Development
Faculty of Organization and Informatics,
University of Zagreb

2. HAI Orchestration Platforms

The challenge of building scalable, robust, large applications that can be incrementally upgraded, yielded a need for a concept called microservices [20]. Within such concept, the core functionality of an application is decomposed into many smaller units, usually packed as containers, that are in some way able to communicate within themselves.

As [31] notes, such applications can be "composed of clusters of hundreds of instances of containerized services", and the cluster of containers must be "fault tolerant, available, and potentially geographically dispersed". Because of the sheer complexity of such systems, the notion of orchestration platforms became highly significant within the domain – such platforms can simplify container management and are generally used for integrating and managing containers at scale. Some of the general container orchestration platforms in use at the time of this writing are Kubernetes, Amazon Elastic Container Service, Docker Swarm, Mesosphere Marathon, RancherOS, etc.

Blanchet et al. [8] introduce an intelligent-agent framework where web services are wrapped in a conversation layer and service orchestration is presented as a conversation among intelligent agents, each of which is responsible for delivering services. Their effort was to avoid inflexible composition evolution where all parties must be updated concurrently in order to avoid interaction problems.

Hahn and Fischer [24] present a model-driven approach to design interoperable holonic multiagent systems in Service-oriented Architectures, illustrating how a choreography model can be easily conceptually implemented through holons.

On the other hand, computer games as applications are usually monolithic with a thick client handling most of the computations. Network based games including massively multiplayer online (MMO) and streamed games only recently shifted their attention to microservices, and there are not many reports on using orchestration platforms for game servers and game system architectures. For example [77] note only three types of system architectures for MMO games: (1) client-server; (2) distributed multi-server and (3) peer to peer. The distributed multi-server architecture is most closely related to the concept of microservices, but not equivalent. According to [77] there are two types of such an architecture: (a) shards - each server contains an own copy of the game world

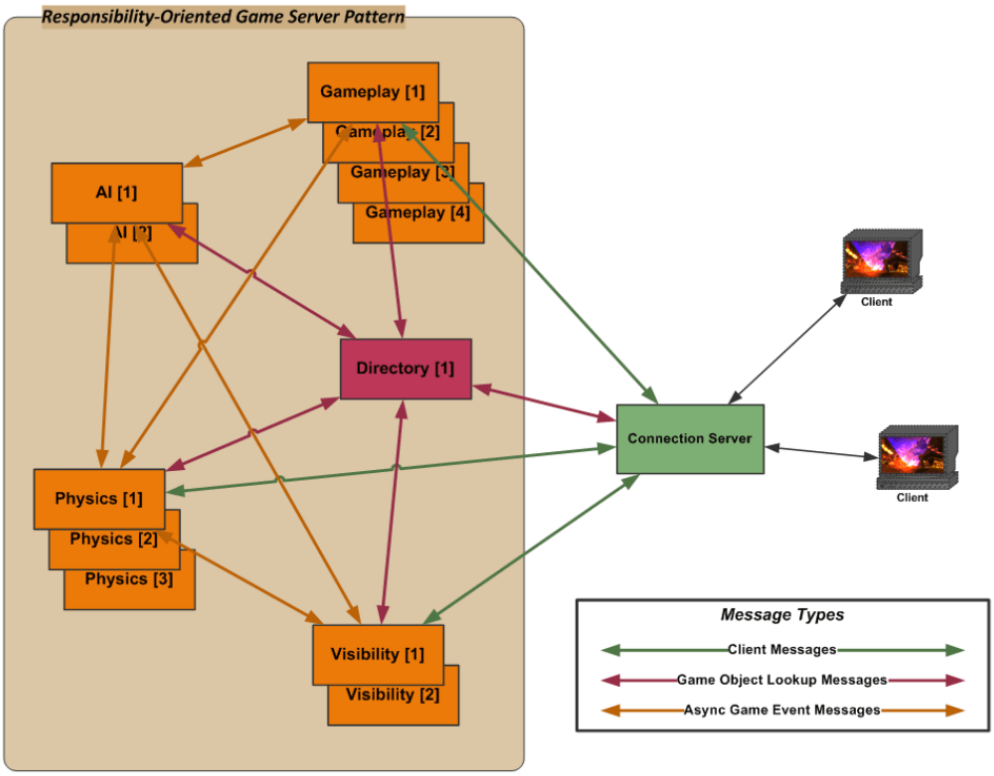


Figure 2.1: Responsibility-Oriented Game Server [69]

and (b) one world - each server contains only one part of the game world. An industry expert [69] mentions even six types of game server architecture patterns:

Monolithic Architecture resembles a server on which the entire game is implemented in a single process on a single computer. First-person-shooters like *Quake 2*, *Unreal Tournament*, *Counter Strike* and *Call of Duty* use such a pattern.

Distributed Network Connections pattern usually implements a server which accepts and manages client connections in order to multiplex client network traffic to several game servers (other names for such a server are connection server, user server, gateway server, front-end server or player server). Most commercial MMO games use this kind of pattern.

Client Side Load Balancing is an implementation pattern in which the game client contains a mechanism to randomly choose between a number of available connection servers described above. Some large-scale MMO game solutions use this kind of pattern.

Map-Centric Game Server [69] describes this pattern as follows: "*Develop a map-centric game server strategy that concentrates core game play activity with the maps in which it occurs. Do this by creating two server types, area server and world server. The game's server cluster contains many area server processes connected to a single central world server process. One or more connection servers generally manage client network connections, but are not part of this pattern.*". From our perspective these area and world server types could be viewed as service types in an orchestrated architecture. Such an implementation pattern is used by *Richard Garriott's Tabula Rasa*, *Hero Engine* (an MMO server framework) and *Face of Mankind*.

Seamless World Game Server is a variant of the previous pattern in which the world map is divided into smaller region maps distributed across a uniform server grid. Such a pattern is

used by games like *Ultima Online*, *Ultima Online 2* (cancelled), the mentioned *Hero Engine* and *Star Wars Galaxies*.

Responsibility-Oriented Game Server is a pattern in which server types responsible for specific subsets of game play functionality (like gameplay, AI, visibility, physics, game state directory etc.) are defined. Such a pattern is used by *Rift (Trion Worlds)* and *Lineage*. This type of architecture closely resembles the idea of a microservices architecture as shown on figure 2.1.

In the following we will argue that orchestration platforms, especially when modeled and implemented as holonic and organization centered multiagent systems (MASs) can profoundly benefit the implementation of game server architectures by introducing various *out-of-the-box* features which are usually manually implemented in MMO game servers.

2.1 Requirements Definition

As already outlined in the introduction, a microservices orchestration platform allows for building scalable, robust, large applications that can be incrementally upgraded [20]. In order to implement a computer game that make use of such an architecture, there are a number of requirements that have to be adhered and a number of features the actual game engine has to offer.

In the following we will describe a cloud service orchestration platform as a holonic multi-agent organisation or holarchy (see [28, 49]). The latter idea, describing services as holons (agent that can be comprised of other agents), provides for the mechanism of nesting services, i.e. making it possible for the platform to run services either as individual services (agents) or complex organizations of agents. In this way it is possible to define multiple levels of holonic organization (see [52] and [51] for details on this approach).

The platform that contains several redundant agents is therefore the back-end of the platform described in this paper. Such agents consist of a chosen process, an input, and an output and can be instanced on demand – usually when load on a given agent becomes to high for it to process it adequately. Holons are, therefore, the structure of the back-end module of the cloud services orchestration platform - they take care of instancing agents when there is need, provide communication and coordination facilities and load balancing. Usually orchestration platforms are *invisible* to the client – the client communicates with a service (an agent in our case) which might be just one such service in an ensemble of agents in the background. Thus, any service that could be offered through the network can be orchestrated in this way, and this includes various aspects of gameplay. Thus, for a game engine in order to communicate with a microservice orchestration platform it needs to feature a simple application programming interfaces (APIs) that is able to communicate with a service needed by the game engine. Most prominent possible APIs are representational state transfer (REST), Web Socket or NetCat (simple TCP or UDP connections) that will enable developers to implement connections between their game and the cloud services orchestration platform. Therefore, in most game development engines all necessary prerequisites are already established since most engines allow for various types of network connections.

On the other hand, in order to make a game that puts an orchestration platform to good use, there are a number of requirements on the actual game architecture. The most important one is modularity – possibly implemented to an actor model (which is basicaly an agent model just in terms of game development usual terminology) in which each actor connects to some kind of service (agent) in order to act upon the game world and react to certain game events. Thus, an agent based approach to implementing both games and their necessary server architecture allows us to handle fairly complex terminology in terms of agents communicating. coordinating, competing etc.

Networked game server architectures including MMO and game streaming services can have major benefits from using a containerized microservices orchestration platform. [31] outline that there are seven key capabilities of a container orchestration platform:(1) cluster state management

and scheduling, (2) providing high availability and fault tolerance, (3) ensuring security, (4) simplifying networking, (5) enabling service discovery, (6) making continuous deployment possible, (7) providing monitoring and governance – all of which would have to be implemented manually when implementing a game server architecture from scratch. Thus, orchestration services provide game backend developers and administrators with a higher-level of abstraction in implementing their desired architecture.

3. Proof-of-concept Examples

In the following we will present six proof-of-concept APIs which were made in five different game engines for a number of games as well as an in-house developed cognitive agent infrastructure. As already stated most general game engines already have the necessary prerequisites for communicating with services. Herein we will show how such communication can be established with four game engines: Godot, RPG Maker, Ren'Py and Blender Game Engine.

3.1 Godot

Godot is a cross-platform, free and open-source game engine that allows for the implementation of both 2D and 3D games [73]. Figure 3.1 shows the main graphical user interface (GUI) of Godot when a game project is loaded.

As an example we have created a simple role-playing game (RPG) like game in which each NPC interacts with the player and gives her/him simple tasks to solve. Figure 3.2 shows the main screen of the game.[1].

Godot can use a number of progamming languages to implement the actual game, but the default language is GDScript. Listing 3.1 shows how communication to a server can be implemented by using the built-in `HTTPRequest` object.

Listing 3.1: Excerpt from game code related to server connection

```
func get_message():
 var data = RQ_RESULT.result
 if data:
  data[ 0 ][ "msg" ] = data[ 0 ][ "msg" ].replace( '\\n', '\n' )
  if data[ 0 ][ "cmd" ] == "giveItem":
   emit_signal("give_item",NAME)
   DONE = true
  return data[ 0 ]
```

[1]The game uses art by Deesiv @ Deviantart, and is based on an example game by Filipe Mice.

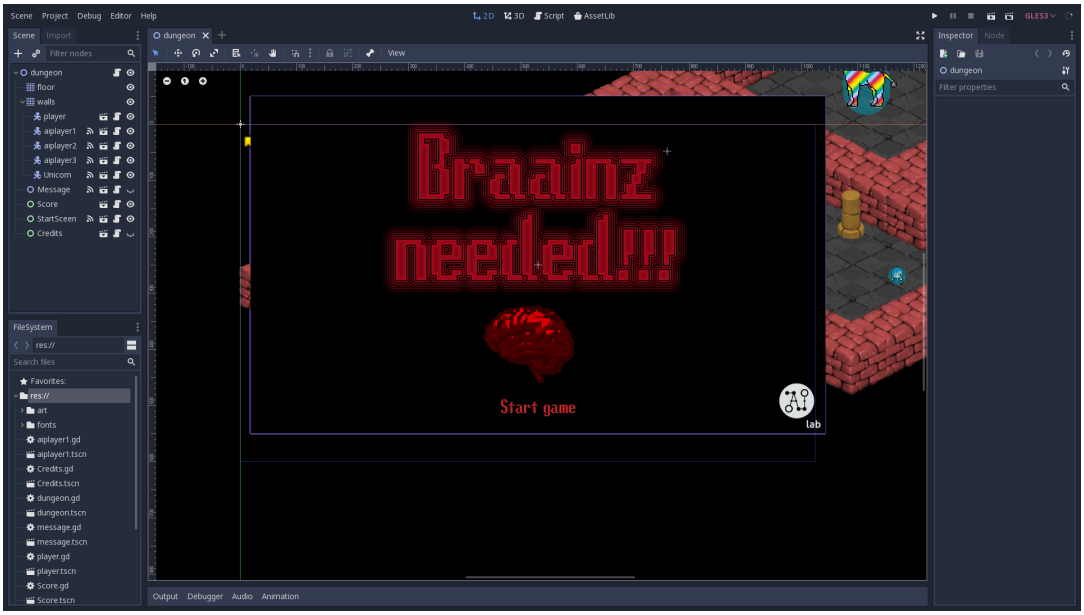


Figure 3.1: Godot – main user interface

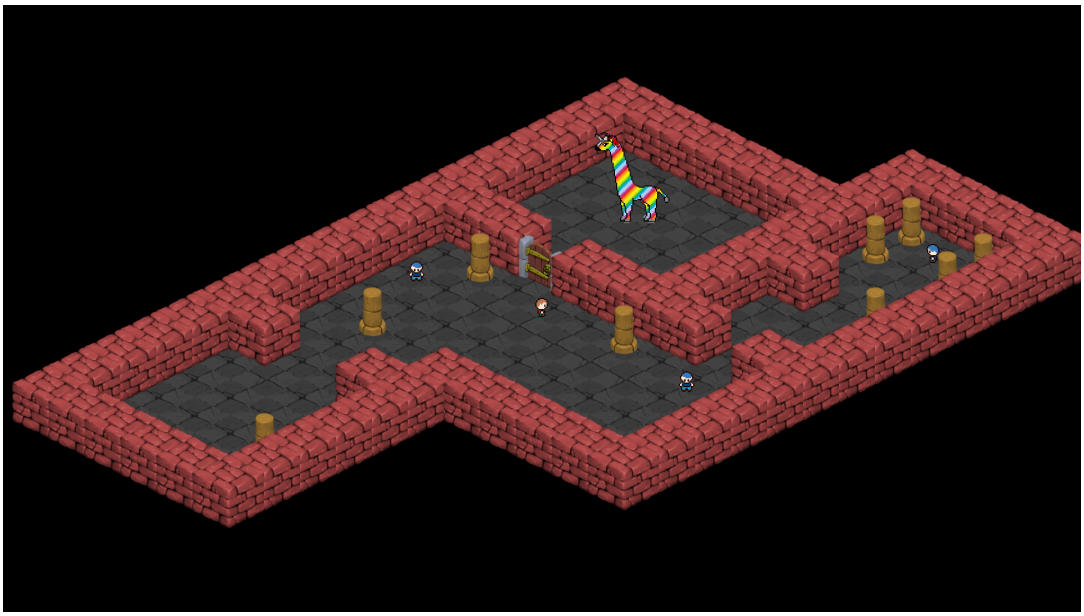


Figure 3.2: Godot – proof-of-concept game *Braainz needed!!!*

```
 return { "msg":"Zombies ate my brain!!!\nCan you program me a new one?\
     ↪ nPretty please ...?", "btns":"Exit", "cmd":"null"}

func send_request( caller ):
 CALLER = caller
 $HTTPRequest.request("http://localhost:2709/query/" + NAME.percent_encode
     ↪ () + "/" + LAST_COMMAND)

func _on_HTTPRequest_request_completed(result, response_code, headers, body
    ↪ ):
```

```
RQ_RESULT = JSON.parse(body.get_string_from_utf8())
CALLER.show_message( self )
```

3.2 RPG Maker

RPG Maker is a tool for creating 2D RPG games [76]. Figure 3.3 shows the main interface of RPG Maker MV.

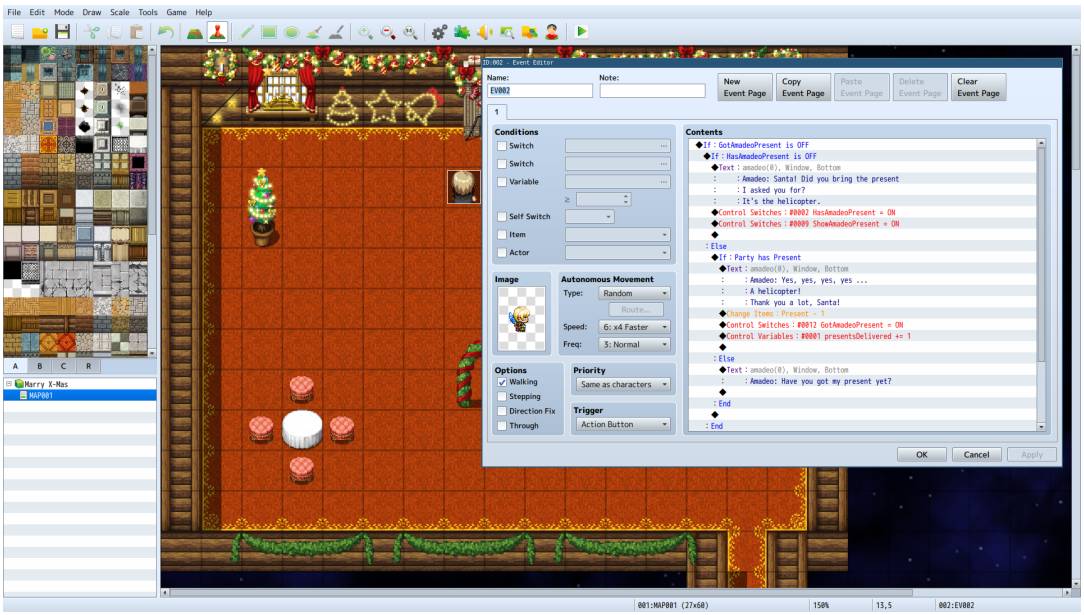


Figure 3.3: RPG Maker – main user interface

In its current version (RPG Maker MV) the scripting language of choice is JavaScript which makes communication to various types of services fairly easy. For this proof-of-concept game we have implemented a simple greeting card game for Christmas shown on figure 3.4.[2]

In order to communicate to a REST service, one can use the built-in `XMLHttpRequest` and `JSON` objects to communicate and parse results respectively, as shown in listing 3.2.

Listing 3.2: Excerpt from game code related to server connection

```
$gameMessage.setFaceImage('Actor1',0);
$gameMessage.setBackground(1);
$gameMessage.setPositionType(2);
let xhr = new XMLHttpRequest();
xhr.open('GET', 'http://localhost:5000/query/ivek', false);
xhr.send();
var ans = JSON.parse( xhr.response )
$gameMessage.add( ans[ 'msg' ] )
```

[2]The game contains a number of game resources avaliable on-line, namely Deep male burp sound by Mike Koenig, Jingle Bells by Kevin MacLeod, Christmas sprites by PandaMaru, Santa face by Idril's Grove, Santa sprite by slimmmeiske2.

Figure 3.4: RPG Maker – proof-of-concept game *Marry X-Mas*

3.3 Ren'Py

Ren'Py is an open source visual novel engine written in Python [75]. The main interface of Ren'Py is shown on figure 3.5.

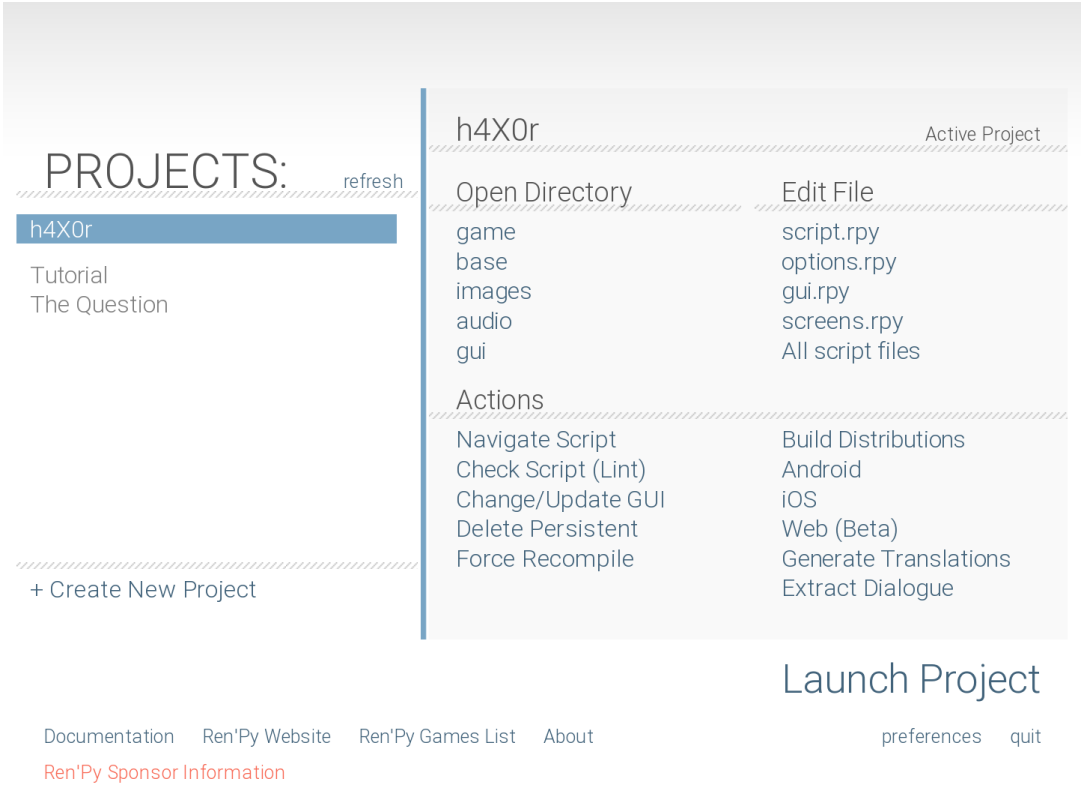


Figure 3.5: Ren'Py – main user interface

As a proof-of-concept game we have implemented a visual novel game (which is still work in progress) shown on figure 3.6.

Figure 3.6: Ren'Py – proof-of-concept game *h4X0r*

Ren'Py uses its *Animation and Translation Language* and its *Screen Language* for game implementation, but it is possible to embed Python code into the script. Listing 3.3 shows an example of how Python code inside Ren'Py script can be used to access a REST service by using Python's `urllib` and `json` modules.

Listing 3.3: Excerpt from game code related to server connection

```
init python:
 import urllib.request, json
 def get_request( req ):
  with urllib.request.urlopen( "http://localhost:8001/q=%s" % req ) as url:
   data = json.loads( url.read().decode() )
   return data

 config.grequest = get_request
```

3.4 Blender Game Engine / UPBGE

Blender Game Engine (BGE) component of Blender, a free and open-source 3D production suite, used for making real-time interactive 3D and 2D games [72]. Since BGE was discontinued, UPBGE (Uchronia Project Blender Game Engine) which is a fork of Blender has been developing the engine further. The main user interface (UI) of UPBGE is shown on figure 3.7.

We have created a small proof-of-concept game in UPBGE shown on figure 3.8.[3]

Since BGE and consequently UPBGE uses Python as its main implementation and scripting language, the implementation of a sensor communicationg with a service is straightforward and similar to the Ren'Py implementation (listing 3.4).

[3]The game uses the Public Domain image of a monkey by frankes.

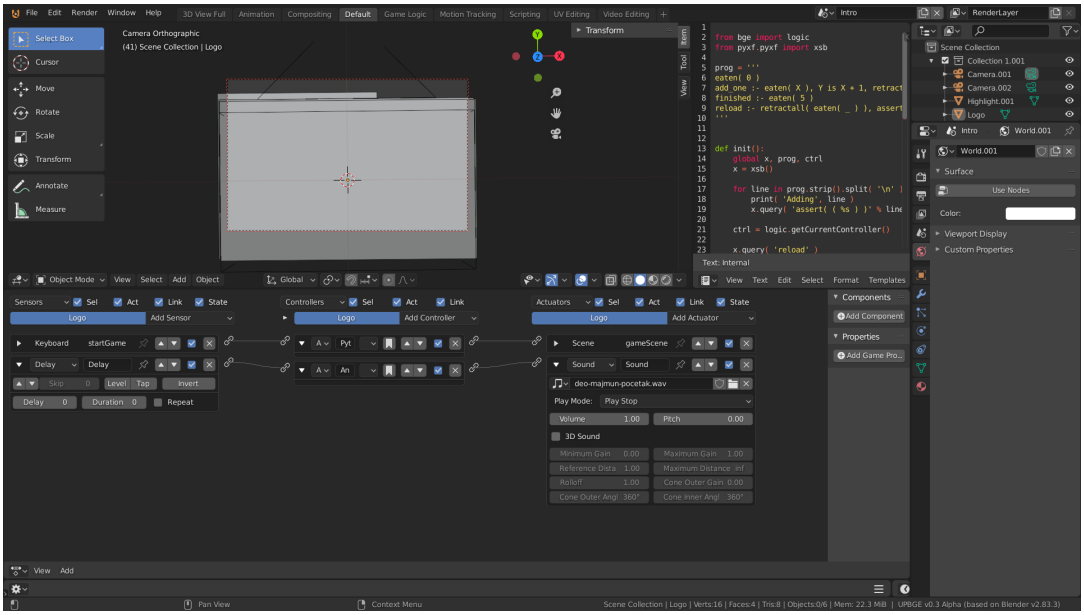


Figure 3.7: Blender Game Engine – main user interface

Figure 3.8: Blender Game Engine – proof-of-concept game *Monkey Doghnut*

Listing 3.4: Excerpt from game code related to knowledge base connection

```
from bge import logic
import urllib.request, json

def query( req ):
 with urllib.request.urlopen( "http://localhost:8001/q=%s" % req ) as url:
  data = json.loads( url.read().decode() )
  return data
```

```
def main():
 if ctrl.sensors[ 'eating' ].positive:
  act = ctrl.actuators[ 'njam' ]
  ctrl.activate( act )
  x.query( 'add_one' )
  result = x.query( 'finished' )
  if result:
   act = ctrl.actuators[ 'finish' ]
   ctrl.activate( act )
```

3.5 Inform 7

3.5.1 Introduction

IF, text adventures, gamebooks and even in some cases visual novels comprise computer games in which players interact with the game using text commands [74]. These narrative worlds usually consist of a number of rooms (whereby the term "room" is very broadly defined and can include any kind of imaginable space or even states of mind) connected by doors (again very broadly defined), and in which objects or things can be placed that can be examined and interacted with. Such things can, for example include NPCs that the player can communicate with, containers that might have other objects within, edibles that can be consumed, wearables that can be used as clothes or equipment, etc. As opposed to most computer games focused on graphics, IF is focused on the story and narrative which makes it an interesting and different medium similarly as printed novels differ from movies.

An important aspect of game design is the integration of AI [78]. Computer games have always been connected to the development of AI. From the earliest chess minmax algorithm by Claude Shannon in 1949 to the more recent AlphaGo™ in 2015, computer games provide an ideal testing environment for AI methods. Similarly, AI has always been an important part of computer games. Computer games have often been judged by the quality of their AI and praised if they used an innovative approach like the ghosts in Pacman™ which had individual personality traits (1980), Creatures™ which used neural networks for character development (1996), Black & White™ which used the BDI model (2000), F.E.A.R.™ which used automated planning algorithms (2005) and many others (see [78, pp. 8–15] for a very detailed overview). Artificial intelligence in games is not only used for NPC or opponent implementation, but also for various other parts of games [78, pp. 151–203] including but not limited to generation of content (graphics including levels and maps, sound, narratives, rules and mechanics or even whole games like the Angelina game-generating system [15]), player behaviour and experience modeling [78, pp. 203–259], as well as bot development and automated game testing [78, pp. 91–151].

Integrating AI into IF seemingly presents a interesting challenge due to specifics of the medium, and there are examples in industry which used various AI techniques in such games, some of which we will address. Herein we present a review on the possibilities AI can provide for IF games and show proof-of-concept applications on how these methods can be applied. In particular we will focus on natural language processing (NLP), autonomous agents and MAS, expert systems as well as content generation.

3.5.2 Interactive Fiction

From computer games stance, IF is a type of a game that is most often a text adventure or a text game [74]. In our example, our focus is set to a text adventure. A such game includes a story and a

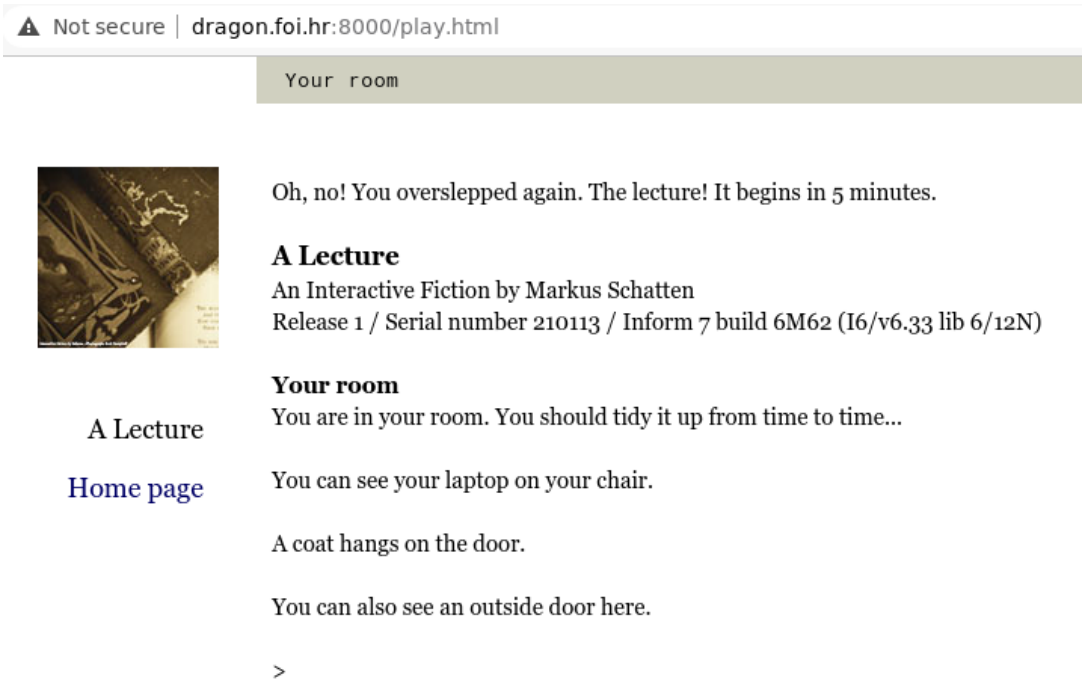


Figure 3.9: A Screenshot of the game *A Lecture* implemented by the authors

narrative that guides a player through it. A player interacts with the game by using text commands. Interaction implies directly to the control of a player's character (for an example, movement) or changes applied to the environment (for an example, collection of an item). The user commands and narrative speech is natural and descriptive enabling players to imagine they are indeed part of a fiction.

Since IF games don't provide a graphical interface, a player needs to rely on its commonsense knowledge and be aware of the environment its character is located at. Another challenge that a player is confronted with are interaction commands. Commands often use natural language, however, there is a limited set of them available to the player. A story genre may also be of a significant importance on how a player progresses through the game [3].

Figure 3.9 previews a sample IF game developed in Inform 7,[4] a declarative programming language for the development of IF based on natural language. In the upper part, we can see narrative output, while in the bottom part there is a player input.

3.5.3 Artificial Intelligence for IF

Being a predominantly textual medium due to text-based input of relevant commands, IF is a relevant application domain for AI-related concepts such as NLP and machine learning (ML). According to the set of select recent publications, which are detailed below, these are being applied to the domain of IF mainly as a means of generating the worlds of those narratives and developing autonomous agents who act as players of developed IF narratives. Furthermore, modern AI methods are used for testing developed narratives [39] and developing dynamic in-game characters [81] as well.

[4] Available here: `http://inform7.com/`

More precisely, AI is, utilising various approaches, used for: authoring instances of open interactive narratives based on crowdsourced example stories [23], making it easier to successfully process natural language player input [16], procedurally generating in-game content and quests [2, 65], and procedurally generating whole worlds founded on knowledge graphs inspired by stories from books [4].

In the context of developing an artificial autonomous player of IF, methods of AI are used for: creating language models, based on a genre-related corpus, suitable for a specific limited game domain [33], implementing knowledge-based players that utilise decision type specifications [25], testing an agent that utilises the conversational learning approach on IF built as an escape room [34], having agents mimic the behaviour demonstrated by human players [48], agents building their knowledge of the world while exploring the game and generating actions based thereon [1], increasing efficiency of generating and evaluating player actions through agents using multi-passage reading comprehension [22], simplifying quick development of player agents [26], building deep reinforcement learning models to be utilised by agents for playing from game-provided feedback alone [30], solving games using model-checkers for C programs [38], and facilitating player performance using semantic parsing [66].

Having IF coupled with AI provides many application opportunities, like adapting the storyline based on the in-game decisions of players [12] or combining elements of IF with Virtual reality (VR) (namely with cinematic VR [47], and artificial storytellers [67]).

A notable example of AI used in combination with IF is AI Dungeon[5], an online text adventure game that utilises the power of GPT-3 [9] since July 2020 to dynamically generate the world wherein the player is located. The generated world is highly dependent on the performed actions of the player, yet the generated story is cohesive, thus providing for a "rich and engaging experience" [36].

To build our examples of integrating AI into IF, we have used Inform 7. As stated earlier, it is a programming language that utilises natural language syntax. The language exposes a broad set of commands covering the most frequent use-cases (for example, some of the commands are: take, go, examine etc.). It also supplies the developer with various concepts to build their narrative. The main concepts we used are: rooms, items, persons and actions.

The beauty of Inform 7 also lays in the fact that to describe a fiction, a developer uses a natural language syntax which motivates the developer to behave and feel as if they were writing a story on a piece of paper. The following code shows a way to describe a world:

```
"The Dungeon" by "Markus Schatten".
Release along with an interpreter.
Release along with a website.
When play begins:
       say "You find yourself in a dungeon surrounded by darkness. The
          ↪ stench is awful." ;
The pit is a room. The description is "This is the place where you woke up.
    ↪  What a scary place!"
A torch is here. The description is "You can see a dim light flickering a
    ↪ few steps away from you."
...
```

In the process of release, a .gblorb file (Glulx story in Blorb format) is generated that is now to be used by an interpreter. An example interpreter session is shown in the following listing:

```
You find yourself in a dungeon surrounded by darkness. The stench is awful.
```

[5] https://play.aidungeon.io/main/landing

```
The Dungeon
An Interactive Fiction by Markus Schatten
Release 1 / Serial number 210114 / Inform 7 build 6M62 (I6/v6.33 lib 6/12N)
pit
This is the place where you woke up. What a scary place!
You can see a torch and a Chest (closed) here.
--> open chest
You open the Chest, revealing an old smelly cheese.
--> take torch and cheese
torch: Taken.
old smelly cheese: Taken.
...
```

3.5.4 Proof-of-Concept Applications

In order to provide some hands-on examples of using various AI techniques with IF we have developed a simple Python interface[6] to the glulxe IF interpreter shell that can execute a number of IF formats in terminal sessions. The developed interface allows us to place filters in front of the IF shell and thus interact with the player on one side and control the game on the other (see fig. 3.10).

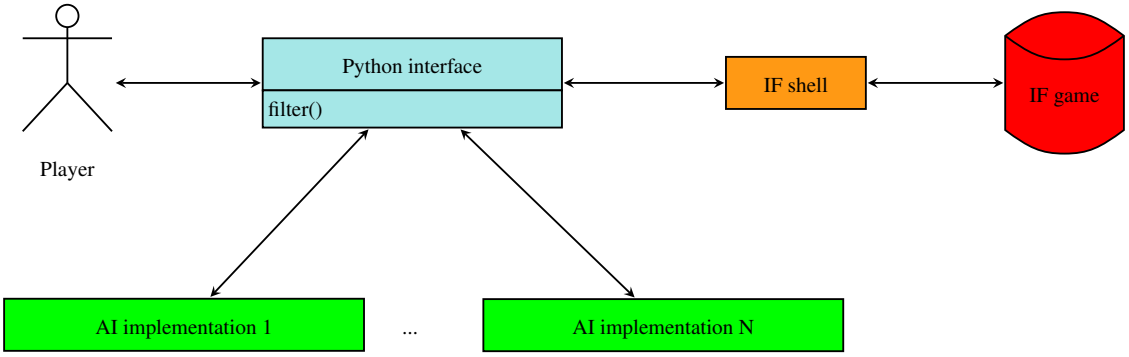


Figure 3.10: Python interface to an IF shell

In the following, we will show four proof-of-concept applications building upon this interface.

3.5.4.1 Natural Language Processing

Whilst NLP has since its beginning been a part of IF, the textual interface to the player has always been constrained to a certain number of commands (in various combinations) that the player could use in order to advance through the narrative. For example, consider the following session excerpt of a IF written in Inform.

```
--> where am i
That's not a verb I recognise.
--> gimme that torch
That's not a verb I recognise.
--> what's in that chest
That's not a verb I recognise.
--> open chest
You open the Chest, revealing an old smelly cheese.
```

[6] Available here: `https://github.com/AILab-FOI/python-glulxe.git`

```
--> take the cheddar
You can't see any such thing.
```

The interpreter can only understand a certain set of predefined commands. Also it does not recognize any possible synonyms of objects or artifacts defined in the game (if they are not explicitly encoded into the game) nor does it recognize common phrases.

Mostly due to recent advancements in ML chatbots [17] have become much more user-friendly than before and numerous tools and services are available for their development. Developing a simple chatbot for a particular purpose has become easy enough for chatbots to enter into widespread usage in numerous domains (like tourism and hospitality for example [45]).

To implement a chatbot filter for the previously mentioned IF game, we have used Chatterbot[7] as can be seen in the following code excerpt.[8]

```
def train( bot ):
    bot.set_trainer( ListTrainer )
    bot.train( [ 'look around', 'look' ] )
    bot.train( [ 'where am i', 'look' ] )
    bot.train( [ 'what is this place', 'look' ] )
    bot.train( [ 'give me that torch', 'take torch' ] )
    bot.train( [ 'i want that torch', 'take torch' ] )
    bot.train( [ 'take that torch', 'take torch' ] )
    bot.train( [ 'what is in that chest', 'open chest' ] )
    bot.train( [ 'let me open that chest', 'open chest' ] )
    bot.train( [ 'yay cheese', 'take cheese' ] )
    bot.train( [ 'i want the cheese', 'take cheese' ] )
    bot.train( [ 'i will make a cheeseburger', 'take cheese' ] )
    bot.train( [ 'take the cheddar', 'take cheese' ] )
    bot.train( [ 'take the gorgonzola', 'take cheese' ] )
...
```

In this way we were able to train the chatbot to understand a number of common phrases that may be used by the player and turn them into the previously mentioned predefined commands. If we now try to play the game using this chatbot as a filter, we are able to use various phrases (and variations thereof since a regression ML model is used) to interact with the narrative, as shown in the following game session:

```
--> where am i
pit
This is the place you woke up. What a scary place!
You can see a torch and a Chest (closed) here.
--> gimme that torch
Taken.
--> what's in that chest
You open the Chest, revealing an old smelly cheese.
--> take the cheddar
Taken.
```

Besides using chatbots as a means of achieving user friendliness of the interface, we could have used it to add additional personality traits to in-game NPCs. For example, we could train one

[7] Available at: `https://chatterbot.readthedocs.io/`

[8] Full example available here `https://github.com/AILab-FOI/python-glulxe/blob/main/examples/dungeon_chatbot.py`

chatbot for each NPC including various special types of conversations that can be understood and performed by each of them. Every time the player interacts with the NPC we could provide control of the shell and likewise conversation to the chatbot thus creating pirates, elfs, orcs, foreigners, secret agents or any imaginable persona to boost player experience.

3.5.4.2 Autonomous Agents

Autonomous agents can provide us with additional dynamics in IF environments. In the following example[9] the goal is to show how IF games could be manipulated by an external agent that randomly generates actions thus directly impacting the game-play regardless of the player's actions. For the implementation we have used SPADE [44] that enables agents to communicate over the Extensible Messaging and Presence Protocol (XMPP).

In this basic example, there are two agents: one that generates actions in random time manner, and another one that interacts with a player and executes game commands (those received from a player or external agent). We have built in two commands in the game that serve to teleport and disarm a player. These commands are not available to the player, but only by the agent. The following listing shows the implementation of these two actions in Inform 7.

```
Disarming is an action applying to nothing. Understand "disarm" as
    ↪ disarming.
Instead of disarming:
        if the player carries anything:
                say "Elf disarmed you";
                now everything carried by the player is in the location;
        otherwise:
                say "Elf tried to disarm you, but you carry nothing"

Teleporting is an action applying to nothing. Understand "teleport" as
    ↪ teleporting.
Instead of teleporting:
        say "Elf teleported you to a different room...";
        move the player to a random room
```

When the autonomous agent decides to interrupt the game by invoking a command, the agent that communicates with the game receives the command and processes it as follows:

```
def teleport_action():
    game.sendline("teleport")
    print_game_response()
```

In the given example, the autonomous agent directly affects the game by invoking game commands. This provides interesting dynamics to the game which is usually static, i.e. can only be changed by user actions or special types of events.

3.5.4.3 Expert Systems

IF games and expert systems (ESs) had a long history together. For example ELIZA [71], one of the first chatbot programs can formally be considered a game of IF since it uses NLP as an interface to the user (player). Lots of ESs had various types of NLP based interpreters very similar to IF shells. With the advancement of IF and reduction of interest in ES this connection was seemingly lost, and most IF platforms like Inform do not have notable facilities that would allow a developer to implement an expert system in-game.

[9] Available here: https://github.com/AILab-FOI/python-glulxe/tree/main/examples/Autonomous%20Agent%20-%20Random%20Actions

Nevertheless, ESs can be of great value to IF game design especially for the implementation of certain "expert" NPCs that can help the player to decide about certain situations. We have developed a very simple decision tree based ES implemented in XSB Prolog[10] that can recognize four types of cheese[11]. By using our interface we can easily allow the player to interact with the ES in-game when (s)he for example asks some NPC (in our example the orc lady) about cheese:

```
import sys, pexpect as px, start
def filter( cmd ):
    if 'ask' in cmd and 'lady' in cmd and 'cheese' in cmd:
       ex = px.spawn( 'xsb --nobanner --quietload --noprompt cheese_expert'
          ↪ )
       ex.interact()
    return cmd
def main( gblorb ):
    start.filter = lambda cmd: filter( cmd )
    start.main( gblorb )
if __name__ == '__main__':
    main( sys.argv[ 1 ] )
```

Note that instead of a simple keyword matching we could have used a chatbot to determine if the player was trying to ask about cheese similarly to the chatbot example given previously. When using this filtered shell we can now obtain a conversation in-game similar to the following:

```
--> ask orc lady about cheese
Orc Lady: Is the cheese soft?
--> no.
Orc Lady: Does is taste very umami?
--> yes.
Orc Lady: Ahh... parmesan, king of all cheeses!
```

3.5.4.4 Generating Content

Although a narrative of an IF instance could be considered similar to a book, and therefore unalterable, the digital context of IF encourages the idea of having parts of such a narrative, or indeed narrative as a whole, generated automatically, as opposed to having been written by a human. Even recent publications (see for example [2, 4, 16, 23, 65, 66]) feature several methods of approaching the problem of generating content for IF.

The approach showcased here is, similar to [4], based on a developed ontology, i.e. a knowledge graph. Such an ontology consists of concepts that can be used to describe the world that should be generated. This description is not final, and contains all the concepts that can be encountered in the proposed world. The concepts existing in the generated world represent a subset of all the concepts that are modelled as available in the observed world. The proof-of-concept example implementation[12] handles the generating process by performing the choice of the concepts to be generated on random. A constraint of the example at this stage is that it can only generate rooms and items possibly with some basic properties within those rooms.

The modelled ontology can be shown visually as a knowledge graph as shown in Fig. 3.11. The individuals visible on Fig. 3.11 are converted to a description of an Inform7 world as follows:

[10] Available at `http://xsb.sourceforge.net/`

[11] Available here: `https://github.com/AILab-FOI/python-glulxe/blob/main/examples/cheese_expert.P`

[12] Available here: `https://github.com/AILab-FOI/python-glulxe/tree/main/examples/Environment%20Generator`

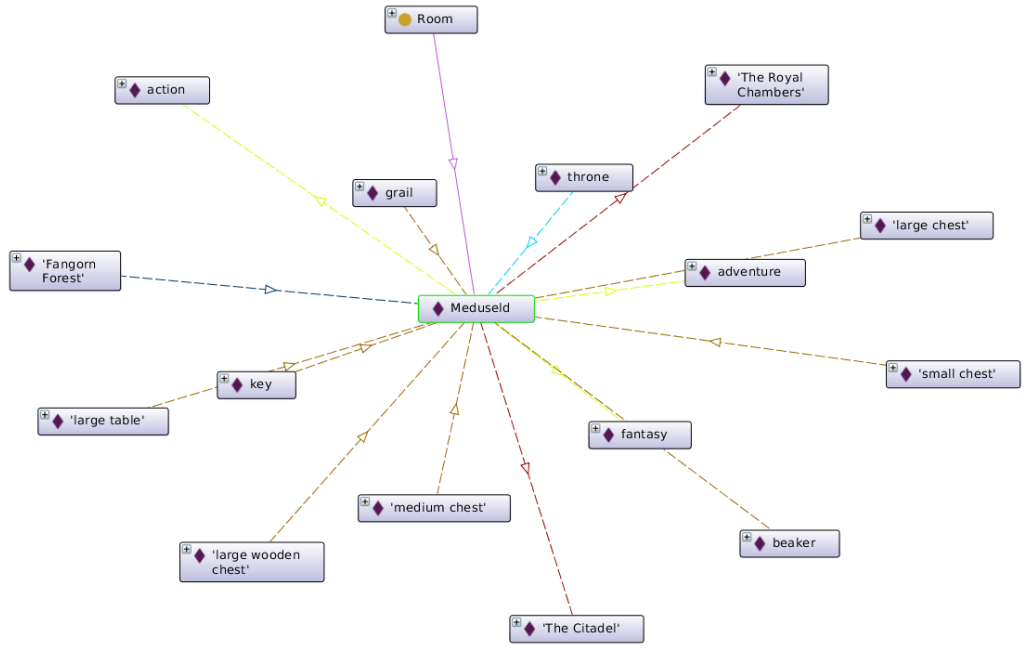


Figure 3.11: Visualised segment of the modelled ontology

```
Meduseld is a room. The description of Meduseld is "You are now in the
    ↪ Golden Hall of Meduseld, the seat of power in Rohan. The king is out,
    ↪  hunting, and there seems to be nobody in the Hall at the moment."
    ↪ Understand "The Golden Hall" as Meduseld.

A metal throne is a thing in Meduseld. The description of the metal throne
    ↪ is "This is the throne of the ruling House of Rohan."
A large chest is a container in Meduseld. The description of the large
    ↪ chest is "A large chest that can house many items." It is opaque and
    ↪  openable.
A medium chest is a container in Meduseld. The description of the medium
    ↪ chest is "A medium chest that can house many items." It is portable
    ↪ and opaque and openable and closed.
A large table is a supporter in Meduseld.
A silver key is a thing on large table. The description of the silver key
    ↪ is "It is not tiny, but does not really fil your palm either." It is
    ↪  portable.
```

3.6 Cognitive Agents

Cognitive agents are a form of artificially intelligent agents that utilize various AI methods including but not limited to ML, NLP, BDI models, knowledge bases (KBs), system automation, speech to text (STT) and text to speech (TTS) in order to allow interaction and learning from humans [37]. There have been many applications of cognitive agents including industrial applications for example based on Internet of things (IoT) and fog computing [19], home service robots [68], cognitive radio [42], mental health therapy [64] and of course education [7].

The implementation of cognitive agents, especially for direct interaction with humans has

always been a challenge. Herein we report on an attempt on implementing a cognitive agent for student university support called Beautiful ARtificial Intelligence Cognitive Agent (BARICA) that has been established at the Faculty of Organization and Informatics, University of Zagreb.

The implemented agent is able to provide students with various information about the city, the faculty building, the staff, working hours, schedules and similar using spoken Croatian language. The agent is aimed to be of help especially to first year students to get to know the faculty and its education process as well as to make the job of administrative staff in the student's office easier. Additionally, BARICA, which is an ongoing open source project,[13] has been subject to study and contribution by students and staff of the Artificial Intelligence Laboratory of the faculty.

Whilst there have been reports about particular aspects of BARICA in media and student works (see for example [63, in Croatian]), this is the first report on the actual architecture of the system, the used technologies as well as the implementation of the casing used for situating the agent in a blended Faculty environment.

3.6.1 Related Work

There have been many attempts of using various AI approaches and particularly chatbots in education systems. Various systems have been built in recent years that take a step forward in how AI enhances the aspects of education. In a systemic review [80] Zawacki-Richter et al. have identified four major application domains: (1) Profiling and prediction (including but not limited to admissions decisions and course scheduling; drop-out and retention; student models and academic achievement); (2) Intelligent tutoring systems (including but not limited to teaching course content; diagnosing strengths and automated feedback; curating learning materials; facilitating collaboration; the teacher's perspective etc.); (3) Assessment and evaluation (including but not limited to automated grading; feedback; evaluation of student understanding, engagement and academic integrity; evaluation of teaching etc.); as well as (4) Adaptive systems and personalisation (including but not limited to teaching course content; recommending personalized content; supporting teachers and learning design; using academic data to monitor and guide students; representation of knowledge in concept maps etc.). The BARICA system falls into the last category using a cognitive chatbot agent interface.

Efforts have been undertaken in profiling and prediction in terms of how likely is it that a student would commit to a programme and won't drop-out. The decision would be based on a student performance such as Grade Point Average (GPA) or its achievements. The output could then be used for teachers to involve more in helping students at risk. In a similar way, some models focus on predicting how successful a student may be in a course based on its engagement [80].

Other kinds of systems are rather focused on having direct impact to a student by providing them with feedback or a content to study. Contrary to prediction and profiling where AI learns based on provided data, in this case systems are involved in communication taking a role of a chatbot. The goal of these systems is to diagnose strengths or gaps in student's knowledge in order to help them accordingly [80].

NLP is also present in the education system that doesn't necessarily involve in communication. The use-case where NLP also has high influence is in automated grading, for example *Automated Essay Scoring* [80].

There is a wide area of use-cases and technologies that improve possibilities of education systems for example: (1) Chatbot [14] that can detect questions posed by students and is able to answer them using NLP techniques and domain ontologies, (2) FIT-EBot [27] a chatbot, which can automatically provide replies to student's question about services provided by the education system on the academic staffs' behalf, (3) FAQBot [46] a chatbot that acts as undergraduate student advisor in the information desk, (4) Chatbot system [13] which models conversations related to

[13] Which is freely available on GitHub here: `https://github.com/AILab-FOI/B.A.R.I.C.A.`

software engineering education, and many others. Herein, due to space contraints, we would like to focus on two chatbot implementations that have the most similarities to the research at hand: Doly, a Bengali chatbot for Bengali education [35] and Smarty [79], a chatbot developed for usage at Zagrebačka škola ekonomije i managementa (ZŠEM).

Doly is a chatbot that serves as a base to build on top of as it provides fundamentals for Bengali language. The primary purpose of it is to be used within the Bengali education systems. The chatbot focuses on properly processing the given input so that it could be used for generating appropriate output [35]. The main AI methods used for the development of Doly are NLP and ML. The development team behind this project used Python technologies to achieve the goals. That being said, to employ NLP, the team used natural language toolkit (NLTK)[14], the conversation logic is built with ChatterBot[15] as well as input adapters. Going forward, ML is used to search predefined corpus of answers and select the most appropriate one for the response to a user [35]. This approach is fairly similar to the approach we have taken in BARICA using almost the same tools, but for the Croatian language and a different approach in selecting appropriate answers: our algorithm is based on a hybrid AI approch combinig numerous regression ML models and finite state machines.

ZŠEM developed Smarty in collaboration with their partners at IBM, a chatbot that answers on frequently asked questions regarding the study at this school. Smarty is based on the IBM Watson Assistant. Some of the main topics that the chatbot is trained to give answers to are: information about study programs, enrollment documentation, mentoring programs and similar. Smarty has been trained to answer questions in Croatian [79]. BARICA on the other hand is not based on any existing chatbot model and has been developed from scratch.

As opposed to both Doly and Smarty, BARICA has a STT and TTS interface including visualization which makes her enact a virtual avatar and thus is much more user-friendly as well as focused on user experience (UX). Also, BARICA is not only a software system, but has also a physical embodiment in form of a specially designed casing which we shall describe in the following sections.

3.6.2 BARICA Architecture

The development of the BARICA system can be broadly divided into two parts: (1) software development and (2) casing / hardware development. The software architecture (as shown on figure 3.12) consists of a cloud-based back-end and an on-site front-end.

The cloud-based back-end is part of a larger framework being developed by the O_HAI (4) Games project (Orchestration of Hybrid Artificial Intelligence Methods for Computer Games) described in more detail in [56, 59]. An initial proof-of-concept implementation of the back-end has also been described in [55]. The back-end comprises a microservice orchestration platfrom based on holonic multi agent systems (HMASs) [50] having (1) a back-end API, (2) a controller and (3) a front-end API. The back-end API allows connecting various microservices including but not limited to knowledge and databases, AI related modules and external services. The controller behaves as microservice orchestration system that allows connecting these various microservices into a coherent system. The front-end API allows for the implementation of front-end applications one of which is the BARICA front-end. The cloud based back-end is implemented mostly in Python. For the sake of BARICA we have used the mentioned NLTK and Chatterbot Python modules to implement the chatbots NLP capabilities for the Croatian language using regression models and finite state machines. For the STT part an external service, namely VoiceNotePad[16], has been used.

[14] Available here: `https://www.nltk.org/`

[15] Available here: `https://chatterbot.readthedocs.io/en/stable/`

[16] Available here: `https://voicenotebook.com/`

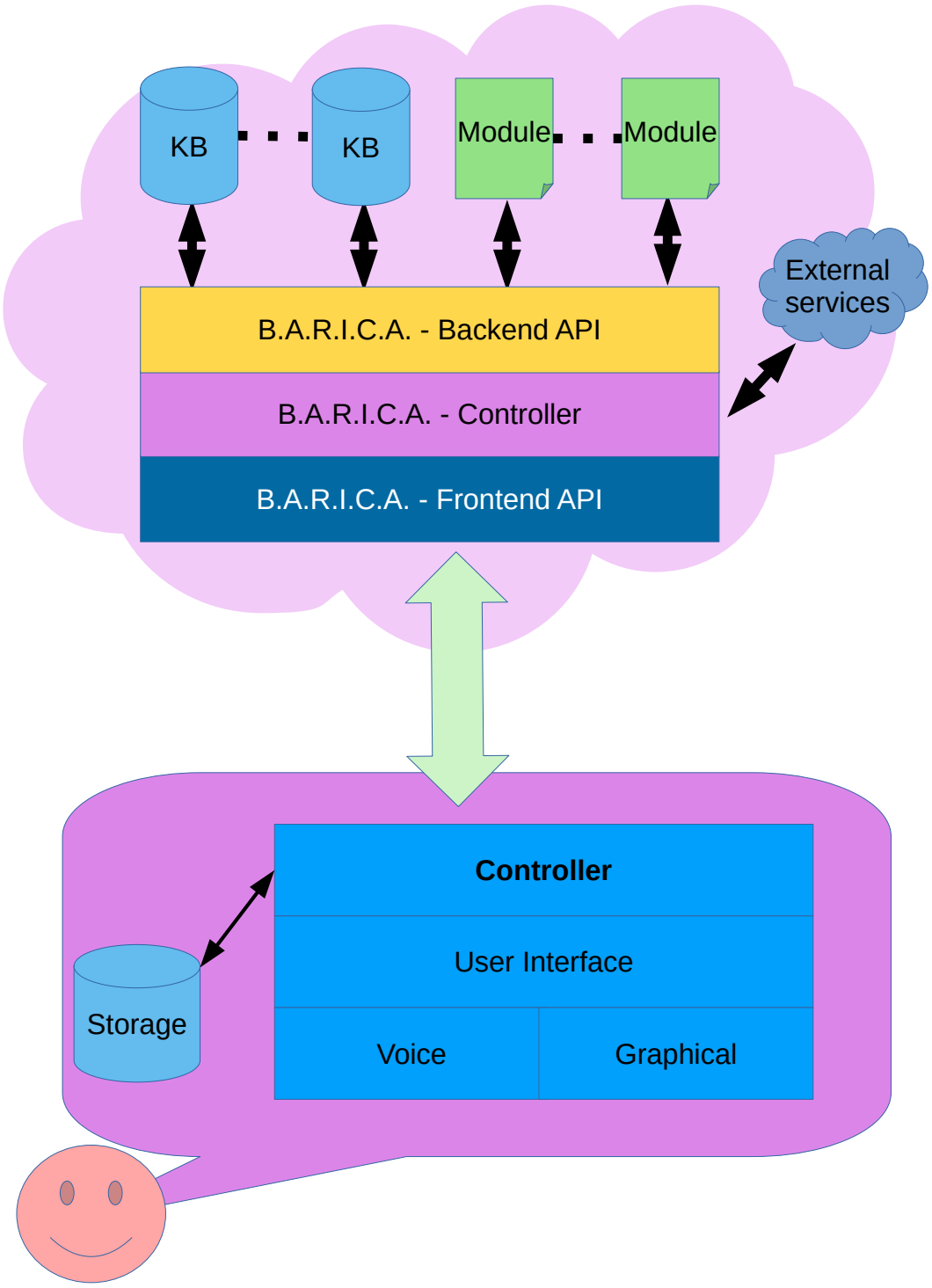


Figure 3.12: BARICA software architecture

For user interface automation we have used PyAutoGUI[17].

The presentation layer, as shown on figure 3.13, has been implemented using Hovercraft[18] with some help of jQuery[19], plain JavaScript and Selenium[20] for browser automation. The STT capabilities have initially been implemented using the speech synthesizer eSpeak[21], but have later been prerecorded for better performance and more natural UX. The lip sync animations have been implemented using CrazyTalk.[22]

The casing / hardware development was a typical do it yourself (DIY) upcycling project that used old furniture (a makeup closet), old electronics (old TV set, old analog telephone), new electronics (flat-screen TV set, PC workstation, microphone etc.) to create a casing that is currently situated in front of the student's office at the Faculty of Organization and Informatics.[23] The whole process of creating the casing has been documented in a digital repository.[24]

Figure 3.13: BARICA presentation layer

Figure 3.14 shows a photograph of the final casing. The system consists of a PC workstation hidden in the makeup closet running Ubuntu Linux, a TV set enclosed into the casing of an old TV set and a microphone hidden in the old telephone's handset which is used for interacting with the system.

[17] Available here: `https://pyautogui.readthedocs.io/`
[18] Available here: `https://hovercraft.readthedocs.io/`
[19] Available here: `https://jquery.com/`
[20] Available here: `https://www.selenium.dev/`
[21] Available here: `http://espeak.sourceforge.net/`
[22] Available here: `https://www.reallusion.com/crazytalk/`
[23] One should mention that there is a good reason why BARICA is developed the way it is. The Faculty of Organization and Infomatics, a modern and state-of-the-art technology oriented research and higher-education institution is situated in the Jesuit and Pauline monastery in Varaždin from 17th century. Similarly BARICA a modern cognitive agent is situated in an old retro-style casing.
[24] Available here: `https://www.dropbox.com/sh/gjmwnf7w3m5dna6/AABM2AZCqN6GxRgXuV9pfXOka?dl=0`

Figure 3.14: BARICA casing as situated at the Faculty of Organization and Informatics

3.6.3 Example Use-Cases

The intended user-base of BARICA are local and visiting students, as well as faculty and staff that may be visiting, or any occasional visitors. This is based on the possible communication flows that are implemented at the present moment, although the possible set of interaction and communication instances can be expanded, and is envisioned to.

According to an insight gained through a series of unstructured interviews with the staff, the faculty, and the students (both local students and incoming exchange students), the most sought-after information resided in the area of lecture hall and laboratory locations, locations of the faculty members' offices, and working hours of members of the faculty. The use-cases described hereafter are based on these scenarios. It should be noted that all the conversation is conducted using spoken Croatian language, both on the side of BARICA, and on the side of the (student) client.

The use-cases described in the following subsections feature a high-level description. The back-end that provides the described functionalities operates in a similar manner for all the three described cases. Before receiving any real spoken commands, BARICA must be initiated, i.e. her listening mode must be started. This is performed by simply voicing her name. Any sound BARICA receives is converted to text, and that resulting string is used in further processing of the input.

Having received the textual version of input through VoiceNotePad, the underlying agent performs a query on their data sources, searching for the input data, i.e. determining whether the input data should be recognised as a specific command, or ignored. If the input data can be found in the database of trained input-output pairs, the accompanying function is retrieved and performed. Based on the results of the performed action, data is either delivered to the user using a pre-rendered animation with audio, or is shown on the screen.

3.6.3.1 Locating a Room

Locating lecture halls and laboratories in the two buildings of the Faculty of Organization and Informatics is no trivial task, especially for the freshmen and people who are not very well acquainted with the layout of the buildings, due to a seemingly confusing numbering of the relevant rooms.

A freshmen, having visited the main building for only a couple of times, may be confused when, for example, late for their lecture. Therefore, they use BARICA located near the main entrance, pick up the phone, and ask her to tell them about the rooms, and where the wanted room is, in simple standard Croatian, for example: "Barice, zanimaju me dvorane." BARICA recognises their intention, and asks them to provide further information, i.e. to tell her the number of the room they want to know more about. In Croatian, the students simply states the number of the lecture hall, e.g. "Dvorana 9." BARICA searches her data sources and delivers the relevant wanted information.

3.6.3.2 Office Hours of the Faculty

More often than not, students need to have a word with members of the faculty, for various reasons. Even more often, it is true that those members of the faculty do not have inexhaustible amounts of free time, and have specified office hours when they are available to talk to students. With this information available online, a student can walk up to BARICA, and ask her to give them information about a specific member of the faculty. In such a case, BARICA opens a browser window, and displays the web page featuring the available, and wanted, information about the specific member of the faculty.

The interaction is, again, simple and concise. First, BARICA is guided by telling her that the person is interested in members of the faculty, in Croatian, for example "Želim informacije o profesorima." BARICA replies with a question demanding more information, i.e. the name of the specific member of the faculty that the person is interested in. Upon receiving the name, the information is delivered in the manner stated above, focusing on the office hours or the location of the office, as per the original request.

3.6.3.3 Information About the Schedule

The longest currently implemented communication flow is tasked with retrieving a student group's schedule. The complexity of this particular use-case is founded in the fact that there are several study programmes being taught at the Faculty of Organization and Informatics, with several orientations each. Each of these study programme orientations features differences in their schedules. Furthermore, students of bachelor-level programmes are usually divide in several study groups, and each group's schedule is slightly different from the others.

Retrieving a specific schedule is, therefore, a modelled decision tree. BARICA guides her users through the necessary decisions using her spoken replies. A member of the staff, or any other end-user, can engage in a conversation with BARICA and ask her to provide them with information on schedules, in Croatian for example "Koji je moj raspored." BARICA guides them through the information necessary for her to provide the initially requested information, i.e. the schedule of a specific study group of a specific study programme's orientation.

3.6.4 Discussion

BARICA is an on-going project that is being developed by members of the AILab and interested students. Having the basic functionalities and adequate APIs in place, it can be extended at will. Thus, there is always room for improvement. BARICA could be extended in different directions and also targeting different audiences. For example, we could also be looking into extending it to produce video materials (either in educational purposes or for motivation). Perhaps, it might be beneficial for some audiences if it could print out an asset like a confirmation or certificate.

Personalizing content based on who BARICA speaks to may also be of high impact. This would need an implementation of an authentication system, like for example, face recognition. Currently, the main targeting audience doesn't go too much outside the scope of institutional personnel, as in, students and teachers, and thus also a simpler authentication systems like voice recognition or 2-factor authentication could be used.

The chatbot has priorly been taught conversations and vocabulary through NLP. However, it is no longer involved in the active learning, therefore, it may end up in situations where it doesn't have proper understanding of what user asks it. It would make sense to involve BARICA in active learning utilizing reinforcement learning strategies. Obviously, it might be challenging to define rewards and penalties for its (in)correct answers. We are currently collecting extensive log data on the usage of the system with hope to use it for such purposes. Extending BARICA to speak different languages may also have a lot of value, especially since more and more students from outside Croatia decide to study at the faculty.

Knowledge of BARICA in its current state is confined to the internal data sources. That makes sense since these data sources are of the most value for the purpose of the chatbot. However, in the future work it could become critical that BARICA starts consuming external data if its capabilities and scope increases.

Another direction of improvement may as well be scaling access of BARICA. The architecture is already put in cloud meaning it has great fundamentals to build upon that to serve many users at the same time. That would mean it could be previewed through any source that provides the required input and output - we would be speaking of mobile phones, tablets, Agumented reality (AR)/VR etc.

Being given the overview of potential improvement directions, we can now speak more about concrete use-cases how to make BARICA even more useful.

By bringing awareness to BARICA in terms of knowing who it speaks to, BARICA may be able to execute tasks for a specific user. From a student's stance, that could be checking its schedule, printing documentation and so on. Looking from a teacher's perspective, that could be executing some of the predefined processes, for example, grading, scheduling exams etc.

If we extend data sources the chatbot consumes, there are numerous of options available. The chatbot could search web on user's behalf and potentially filter results based on the context or criterion - provide results from only confirmed pages. BARICA may also get involved in processes of preparing students for their exams by asking questions and evaluating answers.

4. Conclusion

In this deliverable we have argued that containerized microservices orchestration platforms can benefit game server architectures on a number of ways by allowing to use already established services provided by the orchestration platform. We have shown that both the backend server-side as well as the actual game engine can (and should) be modeled and implemented as a (holonic) multi-agent system. In this way the same simplified and much more human understandable abstraction is used in both sides of the development.

We have also argued that most general game engines already provide the necessary prerequisites for implementing game infrastructures as orchestrated systems, namely: (1) support for actor model (i.e. agent model) and (2) network communication abilities. Other platforms like the BARICA cognitive agent platform in most cases also have all the necessary infrastructure to use microservice orchestration platforms.

In the end we have provided five proof-of-concept games implemented in five different game engines to show how simple it is to implement orchestration server communication from a game engine, since orchestration server behave as simple REST, WebSocket or even Netcat (TCP / UDP) services. Additionally, programming examples have been shown for the interested reader to try out.

Our future research will be focused on implementing valid APIs for a number of game engines to communicate with an orchestration platform focused on HAI services that shall be implemented as part of the O_HAI (4) Games project sponsored by the Croatian Science Foundation.

Bibliography

[1] Prithviraj Ammanabrolu and Matthew Hausknecht. "Graph Constrained Reinforcement Learning for Natural Language Action Spaces". In: *International Conference on Learning Representations*. International Conference on Learning Representations. Virtual, Apr. 27–30, 2020. arXiv: 2001.08837. URL: https://iclr.cc/virtual_2020/poster_B1x6w0EtwH.html (visited on 02/07/2021) (cited on page 17).

[2] Prithviraj Ammanabrolu et al. "Toward Automated Quest Generation in Text-Adventure Games". In: *Proceedings of the 4th Workshop on Computational Creativity in Language Generation*. Workshop on Computational Creativity in Natural Language Generation. Edited by Benjamin Burtenshaw and Enrique Manjavacas. Tokyo, JP: Association for Computational Linguistics, Oct. 29, 2019–Nov. 3, 2020, pages 1–12. URL: https://www.aclweb.org/anthology/2019.ccnlg-1.1 (cited on pages 17, 21).

[3] Prithviraj Ammanabrolu et al. "Bringing Stories Alive: Generating Interactive Fiction Worlds". In: *School of Interactive Computing, Georgia Institute of Technology* (2020), page 3 (cited on page 16).

[4] Prithviraj Ammanabrolu et al. "Bringing Stories Alive: Generating Interactive Fiction Worlds". In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 16.1 (2020), pages 3–9. URL: https://ojs.aaai.org/index.php/AIIDE/article/view/7400 (cited on pages 17, 21).

[5] Danial Jahed Armaghani et al. "Airblast prediction through a hybrid genetic algorithm-ANN model". In: *Neural Computing and Applications* 29.9 (2018), pages 619–629 (cited on page 2).

[6] Aydin Azizi. "Hybrid artificial intelligence optimization technique". In: *Applications of Artificial Intelligence Techniques in Industry 4.0*. Springer, 2019, pages 27–47 (cited on page 2).

[7] Amy Baylor. "Intelligent agents as cognitive tools for education". In: *Educational technology* (1999), pages 36–40 (cited on page 22).

[8] Warren Blanchet, Eleni Stroulia, and Renée Elio. "Supporting adaptive web-service orchestration with an agent conversation framework". In: *IEEE International Conference on Web Services (ICWS'05)*. IEEE. 2005 (cited on page 5).

[9] Tom B. Brown et al. *Language Models Are Few-Shot Learners*. July 22, 2020. arXiv: 2005.14165 [cs]. URL: http://arxiv.org/abs/2005.14165 (visited on 02/07/2021) (cited on page 17).

[10] Kien-Trinh Thi Bui et al. "A novel hybrid artificial intelligent approach based on neural fuzzy inference model and particle swarm optimization for horizontal displacement modeling of hydropower dam". In: *Neural Computing and Applications* 29.12 (2018), pages 1495–1506 (cited on page 2).

[11] Wei Chen et al. "Novel hybrid artificial intelligence approach of bivariate statistical-methods-based kernel logistic regression classifier for landslide susceptibility modeling". In: *Bulletin of Engineering Geology and the Environment* (2018), pages 1–23 (cited on page 2).

[12] Da Hyeon Choi. "LYRA: An Interactive and Interactive Storyteller". In: *2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*. 2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC). New York, NY, USA: IEEE, Aug. 2019, pages 148–153. ISBN: 978-1-72811-664-8. DOI: 10.1109/CSE/EUC.2019.00037 (cited on page 17).

[13] Seo-Won Choi and Jae-Hyun Nam. "The use of AI chatbot as an assistant tool for SW education". In: *Journal of the Korea Institute of Information and Communication Engineering* 23.12 (2019), pages 1693–1699 (cited on page 23).

[14] Fabio Clarizia et al. "Chatbot: An education support system for student". In: *International Symposium on Cyberspace Safety and Security*. Springer. 2018, pages 291–302 (cited on page 23).

[15] Michael Cook, Simon Colton, and Azalea Raad. "Inferring Design Constraints From Game Ruleset Analysis". In: *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE. 2018, pages 1–8 (cited on pages 2, 15).

[16] Margaret Cychosz et al. "Effective Scenario Designs for Free-Text Interactive Fiction". In: *Interactive Storytelling*. Edited by Nuno Nunes, Ian Oakley, and Valentina Nisi. Volume 10690. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pages 12–23. ISBN: 978-3-319-71026-6 978-3-319-71027-3. DOI: 10.1007/978-3-319-71027-3_2 (cited on pages 17, 21).

[17] Robert Dale. "The return of the chatbots". In: *Natural Language Engineering* 22.5 (2016), pages 811–817 (cited on page 19).

[18] Dominik Dellermann et al. "Hybrid intelligence". In: *Business & Information Systems Engineering* (2019), pages 1–7 (cited on page 1).

[19] Fotis Foukalas. "Cognitive IoT platform for fog computing industrial applications". In: *Computers & Electrical Engineering* 87 (2020), page 106770 (cited on page 22).

[20] Martin Fowler and James Lewis. "Microservices a definition of this new architectural term". In: *URL: http://martinfowler. com/articles/microservices. html* (2014), page 22 (cited on pages 5, 7).

[21] Tonglin Fu and Chen Wang. "A hybrid wind speed forecasting method and wind energy resource analysis based on a swarm intelligence optimization algorithm and an artificial intelligence model". In: *Sustainability* 10.11 (2018), page 3913 (cited on page 2).

[22] Xiaoxiao Guo et al. “Interactive Fiction Game Playing as Multi-Paragraph Reading Comprehension with Reinforcement Learning”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Conference on Empirical Methods in Natural Language Processing. Virtual: Association for Computational Linguistics, Nov. 16–20, 2020, pages 7755–7765. ISBN: 978-1-952148-60-6. arXiv: 2010.02386. URL: https://2020.emnlp.org (visited on 02/07/2021) (cited on page 17).

[23] Matthew Guzdial et al. “Crowdsourcing Open Interactive Narrative”. In: *Proceedings of the 10th International Conference on the Foundations of Digital Games*. Foundations of Digital Games. Pacific Grove, CA, USA: Society for the Advancement of the Study of Digital Games, June 22–25, 2015, page 9. ISBN: 978-0-9913982-4-9. URL: http://www.fdg2015.org/proceedings.html (cited on pages 17, 21).

[24] Christian Hahn and Klaus Fischer. “Service composition in holonic multiagent systems: Model-driven choreography and orchestration”. In: *International Conference on Industrial Applications of Holonic and Multi-Agent Systems*. Springer. 2007, pages 47–58 (cited on page 5).

[25] Matthew Hausknecht et al. *NAIL: A General Interactive Fiction Agent*. Feb. 14, 2019. arXiv: 1902.04259 [cs]. URL: http://arxiv.org/abs/1902.04259 (visited on 02/07/2021) (cited on page 17).

[26] Matthew Hausknecht et al. “Interactive Fiction Games: A Colossal Adventure”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.05 (Apr. 3, 2020), pages 7903–7910. ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v34i05.6297 (cited on page 17).

[27] Ho Thao Hien et al. “Intelligent assistants in higher-education environments: the FIT-EBot, a chatbot for administrative and learning support”. In: *Proceedings of the ninth international symposium on information and communication technology*. 2018, pages 69–76 (cited on page 23).

[28] Bryan Horling and Victor Lesser. “A Survey of Multi-Agent Organizational Paradigms”. In: *The Knowledge Engineering Review* 19.04 (Nov. 2005), page 281. ISSN: 0269-8889. DOI: 10.1017/S0269888905000317 (cited on page 7).

[29] Abolfazl Jaafari et al. “Hybrid artificial intelligence models based on a neuro-fuzzy system and metaheuristic optimization algorithms for spatial prediction of wildfire probability”. In: *Agricultural and Forest Meteorology* 266 (2019), pages 198–207 (cited on page 2).

[30] Vishal Jain et al. “Algorithmic Improvements for Deep Reinforcement Learning Applied to Interactive Fiction”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (Apr. 3, 2020), pages 4328–4336. ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v34i04.5857 (cited on page 17).

[31] Asif Khan. “Key characteristics of a container orchestration platform to enable a modern application”. In: *IEEE cloud Computing* 4.5 (2017), pages 42–48 (cited on pages 5, 7).

[32] Mohammadreza Koopialipoor et al. “Applying various hybrid intelligent systems to evaluate and predict slope stability under static and dynamic conditions”. In: *Soft Computing* (2018), pages 1–17 (cited on page 2).

[33] Bartosz Kostka et al. “Text-Based Adventures of the Golovin AI Agent”. In: *2017 IEEE Conference on Computational Intelligence and Games (CIG)* (Aug. 2017), pages 181–188. DOI: 10.1109/CIG.2017.8080433. arXiv: 1705.05637 (cited on page 17).

[34] Cäcilie Kowald and Beate Bruns. "New Learning Scenarios with Chatbots – Conversational Learning with Jix: From Digital Tutors to Serious Interactive Fiction Games". In: *International Journal of Advanced Corporate Learning (iJAC)* 12.2 (Nov. 19, 2019), page 59. ISSN: 1867-5565. DOI: 10.3991/ijac.v12i2.11176 (cited on page 17).

[35] Md Kowsher et al. "Doly: Bengali Chatbot for Bengali Education". In: *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*. IEEE. 2019, pages 1–6 (cited on page 24).

[36] Latitude Team. *AI Dungeon: Dragon Model Upgrade*. Latitude Team. July 14, 2020. URL: https://aidungeon.medium.com/ai-dungeon-dragon-model-upgrade-7e8ea579abfe (visited on 02/08/2021) (cited on page 17).

[37] In Lee. *Encyclopedia of E-business Development and Management in the Global Economy*. IGI Global, 2010 (cited on page 22).

[38] Martin Mariusz Lester. "Program Transformations Enable Verification Tools to Solve Interactive Fiction Games". In: *7th International Workshop on Rewriting Techniques for Program Transformations and Evaluation*. International Workshop on Rewriting Techniques for Program Transformations and Evaluation. Virtual: Université Sorbonne Paris Nord, 2020, page 10 (cited on page 17).

[39] Martin Mariusz Lester. "ScAmPER: Generating Test Suites to Maximise Code Coverage in Interactive Fiction Games". In: *Tests and Proofs*. Edited by Wolfgang Ahrendt and Heike Wehrheim. Volume 12165. Lecture Notes in Computer Science. Cham, CH: Springer International Publishing, 2020, pages 169–179. ISBN: 978-3-030-50994-1. DOI: 10.1007/978-3-030-50995-8_10 (cited on page 16).

[40] Mengshan Li et al. "Prediction of pKa values for neutral and basic drugs based on hybrid artificial intelligence methods". In: *Scientific reports* 8.1 (2018), page 3991 (cited on page 2).

[41] Gunasekaran Manogaran, R Varatharajan, and MK Priyan. "Hybrid recommendation system for heart disease diagnosis based on multiple kernel learning with adaptive neuro-fuzzy inference system". In: *Multimedia tools and applications* 77.4 (2018), pages 4379–4399 (cited on page 2).

[42] JI Mitola. "Cognitive radio. An integrated agent architecture for software defined radio." PhD thesis. Kungliga Tekniska Hogskolan (Sweden), 2002 (cited on page 22).

[43] Bogdan Okreša Đurić. "Organizational Modeling of Large-Scale Multi-Agent Systems with Application to Computer Games". PhD thesis. Faculty of Organization and Informatics, University of Zagreb, 2018 (cited on page 2).

[44] Javier Palanca et al. "SPADE 3: Supporting the New Generation of Multi-Agent Systems". In: *IEEE Access* 8 (2020), pages 182537–182549 (cited on page 20).

[45] Rajasshrie Pillai and Brijesh Sivathanu. "Adoption of AI-based chatbots for hospitality and tourism". In: *International Journal of Contemporary Hospitality Management* (2020) (cited on page 19).

[46] Alaa A Qaffas. "Improvement of Chatbots semantics using wit. ai and word sequence kernel: Education Chatbot as a case study". In: *International Journal of Modern Education and Computer Science* 11.3 (2019), page 16 (cited on page 23).

[47] María Cecilia Reyes and Giuliana Dettori. "Combining Interactive Fiction with Cinematic Virtual Reality". In: *Proceedings of the 9th International Conference on Digital and Interactive Arts*. ARTECH 2019: 9th International Conference on Digital and Interactive Arts. Braga Portugal: ACM, Oct. 23, 2019, pages 1–8. ISBN: 978-1-4503-7250-3. DOI: 10.1145/3359852.3359888 (cited on page 17).

[48] Jessica Rivera-Villicana et al. “Exploring Apprenticeship Learning for Player Modelling in Interactive Narratives”. In: *Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*. CHI PLAY ’19: The Annual Symposium on Computer-Human Interaction in Play. Barcelona Spain: ACM, Oct. 17, 2019, pages 645–652. ISBN: 978-1-4503-6871-1. DOI: 10.1145/3341215.3356314 (cited on page 17).

[49] Sebastian Rodriguez et al. “Holonic Multi-Agent Systems”. In: *Self-Organising Software: From Natural to Artificial Adaptation*. Edited by Giovanna Di Marzo Serugendo, Marie-Pierre Gleizes, and Anthony Karageorgos. Natural Computing Series. Berlin, Heidelberg: Springer, 2011, pages 251–279. ISBN: 978-3-642-17347-9. DOI: 10.1007/978-3-642-17348-6_11 (cited on page 7).

[50] Sebastian Rodriguez et al. “Holonic multi-agent systems”. In: *Self-organising Software*. Springer, 2011, pages 251–279 (cited on page 24).

[51] Markus Schatten. “Reorganization in multi-agent architectures: an active graph grammar approach”. In: *Business Systems Research Journal* 4.1 (2013), pages 14–20 (cited on page 7).

[52] Markus Schatten. “Organizational architectures for large-scale multi-agent systems’ development: an initial ontology”. In: *Distributed Computing and Artificial Intelligence, 11th International Conference*. Springer. 2014, pages 261–268 (cited on page 7).

[53] Markus Schatten, Bogdan Okreša Đurić, and Tomislav Peharda. “A Cognitive Agent for University Student Support (in press)”. In: *2021 IEEE Technology & Engineering Management Conference – Europe*. IEEE. 2021 (cited on page 2).

[54] Markus Schatten, Bogdan Okreša Đurić, and Tomislav Peharda. “Integrating Artificial Intelligence into Interactive Fiction Games – A Review (in review)”. In: *2021 44rd International Convention on Information, Communication and Electronic Technology (MIPRO)*. IEEE. 2021 (cited on page 2).

[55] Markus Schatten, Bogdan Okreša Đurić, and Igor Tomičić. “Towards Simulation of Ambient Intelligence in Autonomous Vehicles using Car Racing Games”. In: *Central European Conference on Information and Intelligent Systems*. Faculty of Organization and Informatics Varazdin. 2019, pages 3–6 (cited on page 24).

[56] Markus Schatten, Bogdan Okreša Đurić, and Igor Tomičić. “Orchestration Platforms for Hybrid Artificial Intelligence in Computer Games-A Conceptual Model”. In: *2020 Central European Conference on Information and Intelligent Systems*. Varaždin, Croatia: Faculty of Organization and Informatics, 2020, pages 3–8 (cited on page 24).

[57] Markus Schatten, Jurica Ševa, and Igor Tomičić. “A roadmap for scalable agent organizations in the internet of everything”. In: *Journal of Systems and Software* 115 (2016), pages 31–41 (cited on page 2).

[58] Markus Schatten, Igor Tomičić, and Bogdan Okreša Đurić. “A review on application domains of large-scale multiagent systems”. In: *Central european conference on information and intelligent systems*. 2017 (cited on page 2).

[59] Markus Schatten, Igor Tomičić, and Bogdan Okreša Đurić. “Towards Application Programming Interfaces for Cloud Services Orchestration Platforms in Computer Games”. In: *2020 Central European Conference on Information and Intelligent Systems*. Varaždin, Croatia: Faculty of Organization and Informatics, 2020, pages 9–14 (cited on pages 2, 24).

[60] Markus Schatten et al. “Agents as bots–an initial attempt towards model-driven mmorpg gameplay”. In: *International conference on practical applications of agents and multi-agent systems*. Springer. 2017, pages 246–258 (cited on page 2).

[61] Markus Schatten et al. “Automated MMORPG Testing–An Agent-Based Approach”. In: *International conference on practical applications of agents and multi-agent systems*. Springer. 2017, pages 359–363 (cited on page 2).

[62] Markus Schatten et al. “Large-Scale Multi-Agent Modelling of Massively Multi-Player On-Line Role-Playing Games–A Summary”. In: *Central European Conference on Information and Intelligent Systems*. 2017 (cited on page 2).

[63] Tajana Šokec. *Modeliranje kontekstualno svjesnog agenta za razgovor na hrvatskom jeziku uz pomoć konačnog automata i strojnog učenja*. Rektorova nagrada Sveučilišta u Zagrebu. Mentor: Markus Schatten. 2019 (cited on page 23).

[64] Shinichiro Suganuma, Daisuke Sakamoto, and Haruhiko Shimoyama. “An embodied conversational agent for unguided internet-based cognitive behavior therapy in preventative mental health: feasibility and acceptability pilot trial”. In: *JMIR mental health* 5.3 (2018), e10454 (cited on page 22).

[65] Adam Summerville et al. “Procedural Content Generation via Machine Learning (PCGML)”. In: *IEEE Transactions on Games* 10.3 (Sept. 2018), pages 257–270. ISSN: 2475-1502, 2475-1510. DOI: 10.1109/TG.2018.2846639 (cited on pages 17, 21).

[66] Ben Swanson and Boris Smus. “Usnea: An Authorship Tool for Interactive Fiction Using Retrieval Based Semantic Parsing”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Annual Meeting of the Association for Computational Linguistics. Online: Association for Computational Linguistics, 2020, pages 263–269. DOI: 10.18653/v1/2020.acl-demos.29 (cited on pages 17, 21).

[67] Sarah Thorne. “Hey Siri, Tell Me a Story: Digital Storytelling and AI Authorship”. In: *Convergence: The International Journal of Research into New Media Technologies* 26.4 (Aug. 2020), pages 808–823. ISSN: 1354-8565, 1748-7382. DOI: 10.1177/1354856520913866 (cited on page 17).

[68] Chien Van Dang et al. “Application of soar cognitive agent based on utilitarian ethics theory for home service robots”. In: *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE. 2017, pages 155–158 (cited on page 22).

[69] Matthew Walker. *Game Server Architecture Patterns*. https://gameserverarchitecture.com/game-server-architecture-patterns/. [Online; accessed 15-July-2020]. 2020 (cited on page 6).

[70] Minggang Wang et al. “A novel hybrid method of forecasting crude oil prices using complex network science and artificial intelligence algorithms”. In: *Applied energy* 220 (2018), pages 480–495 (cited on page 2).

[71] Joseph Weizenbaum. “ELIZA—a computer program for the study of natural language communication between man and machine”. In: *Communications of the ACM* 9.1 (1966), pages 36–45 (cited on page 20).

[72] Wikipedia contributors. *Blender Game Engine — Wikipedia, The Free Encyclopedia*. [Online; accessed 17-July-2020]. 2020. URL: https://en.wikipedia.org/w/index.php?title=Blender_Game_Engine&oldid=967025085 (cited on page 13).

[73] Wikipedia contributors. *Godot (game engine) — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Godot_(game_engine)&oldid=967607161. [Online; accessed 15-July-2020]. 2020 (cited on page 9).

[74] Wikipedia contributors. *Interactive fiction — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Interactive_fiction&oldid=996857758. [Online; accessed 2-February-2021]. 2020 (cited on page 15).

[75] Wikipedia contributors. *Ren'Py — Wikipedia, The Free Encyclopedia*. [Online; accessed 15-July-2020]. 2020. URL: https://en.wikipedia.org/w/index.php?title=Ren%27Py&oldid=960243958 (cited on page 12).

[76] Wikipedia contributors. *RPG Maker — Wikipedia, The Free Encyclopedia*. [Online; accessed 15-July-2020]. 2020. URL: https://en.wikipedia.org/w/index.php?title=RPG_Maker&oldid=967986970 (cited on page 11).

[77] Amir Yahyavi and Bettina Kemme. "Peer-to-peer architectures for massively multiplayer online games: A survey". In: *ACM Computing Surveys (CSUR)* 46.1 (2013), pages 1–51 (cited on page 5).

[78] Georgios N Yannakakis and Julian Togelius. *Artificial intelligence and games*. Volume 2. Springer, 2018 (cited on pages 2, 15).

[79] Zagreb School of Economics and Management. *Virtual assistant at ZSEM*. 2018. URL: https://www.zsem.hr/en/news/virtual-assistant-at-zsem/. (accessed: 20.2.2021) (cited on page 24).

[80] Olaf Zawacki-Richter et al. "Systematic Review of Research on Artificial Intelligence Applications in Higher Education – Where Are the Educators?" In: *International Journal of Educational Technology in Higher Education* 16.1 (Dec. 2019), page 39. ISSN: 2365-9440. DOI: 10.1186/s41239-019-0171-0 (cited on page 23).

[81] Manyu Zhang. "Application of Artificial Intelligence Interactive Storytelling in Animated". In: *2020 International Conference on Control, Robotics and Intelligent System*. CCRIS 2020: 2020 International Conference on Control, Robotics and Intelligent System. Xiamen China: ACM, Oct. 27, 2020, pages 37–41. ISBN: 978-1-4503-8805-4. DOI: 10.1145/3437802.3437809 (cited on page 16).