SAARLAND UNIVERSITY

Faculty of Mathematics and Computer Science
Department of Computer Science
MASTER THESIS

MAJAN: Multiagent Coordination with AJAN for Mobile Cloud Application MyWELCOME

submitted by
Akbar Kazimov
Saarbrücken
July 7, 2022

Advisor:
PD Dr. Matthias Klusch
German Research Center for Artificial Intelligence
Saarland Informatics Campus
Saarbrücken, Germany

Co-Advisor:
M.Sc. André Antakli, M.Sc. Daniel Spieldenner
German Research Center for Artificial Intelligence
Saarland Informatics Campus
Saarbrücken, Germany

Reviewer 1: PD Dr. Matthias Klusch
Reviewer 2: Prof. Dr.-Ing. Philipp Slusallek

Saarland University
Faculty MI – Mathematics and Computer Science
Department of Computer Science
Campus - Building E1.1
66123 Saarbrücken
Germany

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, _____ _____

(Datum/Date) (Unterschrift/Signature)

Abstract

Coordination of actions and forming of groups of agents are key problems in multiagent systems. AJAN is an agent engineering tool that supports SPARQL Behavior Tree-based (SPARQL-BT) agent modeling where SPARQL queries are used to access agent knowledge. However, AJAN does not support the coordination and forming of groups of agents yet.

In this thesis, I investigate and develop a novel approach, MAJAN, to support AJAN users in the realization and evaluation of multiagent coordination use-cases. In MAJAN, agents coordinate in a distributed way based on their SPARQL-BTs unlike VGI where BT-based action coordination is supported but behaviors of agents are managed by a central BT. Moreover, a dedicated agent forms groups based on the collected information in MAJAN. For this purpose, state-of-the-art techniques are leveraged for clustering and coalition structure generation. For developing MAJAN, a user-centered development technique is used to extend the existing AJAN features. More specifically, this work conducts surveys with potential users, collects requirements, designs concepts and UI mockups, and implements them to provide the required features. As the use-cases for MAJAN, Language Course Coordination (LCC) and Co-Habitation Coordination (CHC) scenarios of the mobile cloud-based application My-WELCOME that supports refugee integration to Europe, are realized and evaluated. Since the use-cases require the constraint-based forming of refugee groups and each refugee is provided with a personalized My-WELCOME agent, the agents need to coordinate themselves into optimal groups with MAJAN.

According to the evaluation of the system based on the System Usability Scale (SUS), the performance of MAJAN is *excellent*. Moreover, based on the evaluation of the results of LCC and CHC use-cases, their users are *very satisfied* and *satisfied*, respectively.

Contents

Declaration **ii**

Abstract **iii**

1 Introduction **2**
1.1 Motivation . . . 2
1.2 Goals and Approach . . . 3
1.3 Outline . . . 3

2 Background and Related Work **5**
2.1 Agent Engineering Tool AJAN . . . 5
2.1.1 Agents and Multiagent Systems . . . 5
2.1.2 RDF and SPARQL . . . 6
2.1.3 Behavior Trees . . . 8
2.1.4 AJAN Architecture . . . 10
2.1.5 AJAN Implementation . . . 12
2.2 Mobile Cloud Application MyWELCOME . . . 13
2.2.1 Mobile Cloud Computing . . . 13
2.2.2 MyWELCOME Application . . . 14
2.2.3 MyWELCOME Multiagent Coordination Use-Cases . . . 17
2.3 Coordination in Multiagent Systems . . . 20
2.3.1 Classes of Grouping Algorithms . . . 21
2.3.2 Coordination in Multiagent Engineering Tools . . . 25
2.3.3 Summary . . . 28

3 MAJAN: Multiagent Coordination with AJAN **31**
3.1 Development Methodology . . . 31
3.1.1 Overview . . . 31
3.1.2 Requirements Phase . . . 34
3.1.3 Design Phase . . . 35
3.1.4 Implementation Phase . . . 35
3.1.5 Evaluation Phase . . . 35
3.2 Development Results . . . 37
3.2.1 Initial Concepts . . . 37
3.2.2 Elicited Requirements . . . 40
3.2.3 Designed Concepts . . . 41
3.2.4 Implementation of Features . . . 49
3.2.5 System Usability Evaluation . . . 59
3.2.6 Final Version of MAJAN . . . 60

4 Language Course Coordination Use-Case **62**
4.1 LCC Description . . . 62
4.2 LCC Conceptual Solution . . . 63

4.2.1 Analysis of the Grouping Algorithms for LCC 63
4.2.2 LCC Coordination Protocol 68
4.2.3 LCC Coordination Algorithms and Assumptions 71
4.3 LCC Implementation 74
4.4 LCC Evaluation 85
4.4.1 Settings 85
4.4.2 Results 87

5 Co-Habitation Coordination Use-Case 89
5.1 CHC Description 89
5.2 CHC Conceptual Solution 90
5.2.1 Analysis of Grouping Algorithms for CHC 90
5.2.2 CHC Coordination Protocol 93
5.2.3 CHC Coordination Algorithms and Assumptions 95
5.3 CHC Implementation 97
5.4 CHC Evaluation 101
5.4.1 Settings 101
5.4.2 Results 103

6 Conclusion 105
6.1 Summary 105
6.2 Limitations and Future Work 106

Bibliography 108

A MAJAN User Requirements Survey 114
A.1 User Context Questions 114
A.2 Multiagent Coordination Use-Case Realization Questions 115
A.3 Multiagent Coordination Use-Case Monitoring and Evaluation Questions 122
A.4 MAJAN Guideline Questions 128

B MAJAN Usability Evaluation Survey 130
B.1 User Background Questions 130
B.2 Tasks 130
B.3 System Usability Scale Questions 132
B.4 Complementary Questions 132

C LCC Data Records And Results 134
C.1 Data Records 134
C.2 Results 140

D CHC Data Records And Results 151
D.1 Data Records 151
D.2 Results 159

List of Tables

2.1 Possible node types of RDF components 6
2.2 Example RDF triples 7
2.3 Example RDF triples 7
2.4 The types of control flow and execution nodes [27] 9
3.1 SUS score interpretation according to [45] 60
4.1 Likert scale [22, 45] 85
4.2 LCC data record 1 86
5.1 Default importance weights of CHC individual preferences 90
5.2 CHC data record 1: Demographic data of TCNs 102
5.3 CHC data record 1: Individual Preferences of TCNs 102
5.4 Second configuration for CHC 103
5.5 The third configuration for CHC 103
C.1 LCC data record 2 134
C.2 LCC data record 3 135
C.3 LCC data record 4 135
C.4 LCC data record 5 136
C.5 LCC data record 6 136
C.6 LCC data record 7 137
C.7 LCC data record 8 137
C.8 LCC data record 9 138
C.9 LCC data record 10 139
C.10 LCC data record 11 140
D.1 CHC data record 2: Demographic data of TCNs 152
D.2 CHC data record 2: Individual Preferences of TCNs 152
D.3 CHC data record 3: Demographic data of TCNs 153
D.4 CHC data record 3: Individual Preferences of TCNs 153
D.5 CHC data record 4: Demographic data of TCNs 154
D.6 CHC data record 4: Individual Preferences of TCNs 155
D.7 CHC data record 5: Demographic data of TCNs 157
D.8 CHC data record 5: Individual Preferences of TCNs 158

List of Figures

2.1 Example SPARQL query . . . 8
2.2 Example Behavior Tree . . . 9
2.3 AJAN Architecture [10] . . . 10
2.4 An example BT in AJAN Editor . . . 11
2.5 Mobile Cloud Computing Architecture. [62] . . . 13
2.6 WELCOME technical architecture from the perspective of coordination use-cases . . . 15
2.7 MAJAN comparison table . . . 29
3.1 Waterfall Model [49] . . . 32
3.2 Star model [49] . . . 32
3.3 The life cycle model suggested by [49] . . . 33
3.4 The selected life cycle model for the development of MAJAN [49] . . . 34
3.5 UI mockup of the Monitoring functionality . . . 38
3.6 UI mockup of the Evaluation functionality . . . 39
3.7 FIPA Request Protocol Diagram with BT Labels . . . 42
3.8 BT_1 sends a request to participants . . . 43
3.9 BT_2 receives a request and returns an agreement or a refusal . . . 43
3.10 BT_3 receives and stores the responses of participants . . . 44
3.11 BT_4 sends the result of request to initiator agent . . . 44
3.12 CSGP Coordination Protocol Diagram . . . 45
3.13 Clustering Coordination Protocol Diagram . . . 47
3.14 The updated version of Evaluation Panel mockup . . . 48
3.15 Behaviors tab in AJAN Editor to model behaviors of agents . . . 50
3.16 Send Coordination Request BT of Request Coordination protocol . . . 51
3.17 Handle Coordination Request BT of Request Coordination protocol . . . 52
3.18 Save Coordination Response BT of Request Coordination protocol . . . 52
3.19 Implemented Monitoring panel in MAJAN . . . 53
3.20 Common Structure of MAJAN Logs . . . 54
3.21 The implemented version of the Evaluation panel in MAJAN . . . 54
3.22 MAC Solution section in MAJAN . . . 55
3.23 The format to store a MAC problem and its solutions in AKB . . . 56
3.24 Central Solution section in MAJAN . . . 57
3.25 The Compare Solutions section in MAJAN . . . 58
3.26 Mapping file structure to compute NMI score . . . 59
3.27 Final version of Monitoring panel in MAJAN . . . 61
3.28 Final version of Evaluation panel in MAJAN . . . 61
4.1 Comparison of Grouping algorithms for LCC . . . 64
4.2 LCC interaction diagram between the WELCOME system components . . . 68
4.3 LCC coordination protocol among agents . . . 69
4.4 SPARQL Construct query to execute BOSS algorithm as a BT node . . . 71
4.5 Start LCC Assessment phase BT . . . 74
4.6 Complete LCC Assessment phase BT . . . 75
4.7 The initiating BT of the LCC protocol . . . 75

4.8 Compute CSGP Coalitions Behavior Tree 76
4.9 SPARQL Update query to apply attendance constraint (2d) of LCC 76
4.10 Handle CSGP Utilities BT 77
4.11 SPARQL Construct query to apply individual preferences when computing the utility values for LCC 78
4.12 SPARQL Update query to compute coalition values for LCC 79
4.13 Solve CSGP BT 80
4.14 Start Assessment feature in TP 81
4.15 Assessment Notification in the MyWELCOME application 82
4.16 Assessment exercise types in the MyWELCOME application 82
4.17 Assessment questions in the MyWELCOME application 83
4.18 Start coordination feature in TP 83
4.19 Assign preferences in the MyWELCOME application 84
4.20 Visualization of the solutions in TP 84
5.1 Comparison of Grouping algorithms for CHC 91
5.2 CHC Coordination Protocol 94
5.3 *Start CHC Protocol* BT covers steps 1, 2 and 3 of CHC protocol (figure 5.2) . 97
5.4 *Start Clustering Protocol* BT covers step 4 of CHC protocol (figure 5.2) . . . 98
5.5 *Return Profile Information* BT covers steps 5 and 6 of CHC protocol (figure 5.2) 98
5.6 *Collect Clustering Distances* BT covers step 7 of CHC protocol (figure 5.2) . 99
5.7 *Compute Clustering Distances* BT covers steps 8 and 9 of CHC protocol (figure 5.2) 100
5.8 *Solve Clustering* BT covers the steps 10 and 11 of CHC protocol (figure 5.2) 100
5.9 *Finalize Coordination* BT covers the steps 12 and 13 of CHC protocol (figure 5.2) 101
A.1 Modelling BTs in AJAN Editor 117
A.2 Representation of template BTs in AJAN Editor 118
A.3 Create Agent in AJAN Editor 119
A.4 Send a message to an endpoint of an agent in AJAN Editor 120
A.5 Selection of a single agent 121
A.6 Selection of multiple agents. One-to-Many feature 121
A.7 Monitoring Panel 123
A.8 MAC Solution section 124
A.9 Baseline Solution section 125
A.10 Comparison section 126
A.11 Alternative Chart 1 126
A.12 Alternative Chart 2 127
A.13 Alternative Chart 3 127
A.14 Alternative Chart 4 128
A.15 Workflow for the Expert and Non-expert users to realize MAC Use-Cases 129

Acronyms

ACE Adaptive Clustering Ensemble. 23, 65, 91

ACL Agent Communication Language. 21, 26, 27

ACS Agent Code-based Coalition Structure Search. 24, 37, 65, 92

AI Artificial Intelligence. 8

AJAN Accessible JAVA Agent Nucleus. iii, viii, 2, 3, 5, 10–12, 16, 21, 25, 34–38, 40, 41, 59, 65, 105, 106, 114–117

AKB Agent Knowledge Base. vii, 51, 52, 56

API Application Programming Interface. 12

BCOP-KMeans Boosted Constrained K-Means Clustering. 22, 65, 91

BDI Belief-Desire-Intent. 11, 25, 26

BOSS Bi-directional Search Algorithm for Optimal Coalition Structure Generation. vii, 24, 37, 65, 70, 71, 92

BT Behavior Tree. iii, vii, viii, 2, 3, 5, 8, 9, 11, 12, 16, 25–30, 37, 38, 40–46, 49–53, 71, 74, 75, 77, 80, 93, 97–101, 105–107

CHC Co-Habitation Coordination. iii, vi, viii, 3, 14, 17, 19, 35, 89–103, 106, 151–155, 157–159

CMD Command Prompt. 57

CoExDBSCAN Density-based Clustering with Constrained Expansion. 22, 23, 65, 91

COP-KMeans Constrained K-Means Clustering. 22, 64, 91

CPL Course Progress Level. 17–19, 62, 63, 67, 85, 86, 134–140

CSAT Customer Satisfaction Score. 85, 87, 88, 101, 103, 104, 140–150, 159–163

CSGP Coalition Structure Generation Problem. 2, 24, 37, 45–47, 49, 50, 59, 65, 67, 70, 74, 75, 79, 80, 106

CSS Cascading Style Sheets. 49

DBSCAN Density-based Spatial Clustering of Applications with Noise. 22, 23, 37, 65, 91

DFKI Deutsches Forschungszentrum für Künstliche Intelligenz. 2, 105, 114

FIPA Foundation for Intelligent Physical Agents. 2, 3, 21, 25–28, 30, 37, 40, 42, 49, 50, 106

FSM Finite State Machine. 8

Group Formation RRS Group Formation Reciprocal Recommender. 25

GUI Graphic User Interface. 26, 30

HDBSCAN* Hierarchical Density-based Clustering. 23, 37, 65, 92

HTML HyperText Markup Language. 49

HTTP HyperText Transfer Protocol. 10

IDP Improved Dynamic Programming. 24

IP Integer Programming. 24

JSON JavaScript Object Notation. 56

KQML Knowledge Query and Manipulation Language. 26

LAKR Local Agent Knowledge Repository. 16

LAR Local Agent Repository. 16, 17

LCC Language Course Coordination. iii, vi–viii, 3, 14, 16, 17, 35, 62–80, 85–87, 101, 106, 134–140

MAC Multiagent Coordination. vii, viii, 2, 3, 5, 8, 25–28, 30, 34, 35, 37, 39–41, 51, 54–56, 59, 105–107, 115, 116, 118–125, 128, 129

MAJAN Multiagent Coordination in AJAN. 5, 34, 35

MAS Multiagent System. 2, 3, 6, 20, 27, 37, 40, 105

MCC Mobile Cloud Computing. 13

NMI Normalized Mutual Information. vii, 39, 59, 125

PC Personal Computer. 35

RAM Random Access Memory. 12

RDF Resource Description Framework. 2, 3, 5–8, 10, 12, 16, 25, 39–41, 105

RDF4J Resource Description Framework for JAVA. 10, 12, 16

REST Representational State Transfer. 10, 16

RRS Reciprocal Recommender System. 24, 25, 92

SD Standard Deviation. 67, 79, 80

SDLC Software Development Life Cycle. 32, 105, 106

SPARQL SPARQL Protocol and RDF Query Language. iii, 2, 3, 5, 7, 8, 11, 12, 16, 25, 27, 28, 39, 40, 70, 80, 105

SPARQL-BT SPARQL-BT. iii, 2, 3, 5, 10–12, 21, 25, 29, 30, 34, 36, 41, 105, 106

SUS System Usability Scale. iii, 36, 59, 60, 106, 132

TCN Third Country National. vi, 14–20, 62, 63, 66, 67, 69, 73, 80, 81, 83–86, 89, 90, 92, 93, 95, 96, 101–103, 134, 135, 152–155, 158

TP Teacher Panel. viii, 63, 69–72, 74, 80, 81, 83, 84

UI User Interface. iii, vii, 3, 16, 34–41, 49, 53, 58, 60, 80, 83, 101, 106

URI Uniform Resource Identifier. 6

URL Uniform Resource Locator. 16

VAC Visual Analytics Component. 16

VGI Video Game Industry. iii

W3C World Wide Web Consortium. 6, 7

WAR Welcome Agent Repository. 16, 17, 71

WATR Welcome Agent Template Repository. 16

WBTTR Welcome Behavior Tree Template Repository. 16

WPM Welcome Platform Manager. 16, 17, 71

Chapter 1
Introduction

1.1 Motivation

Intelligent Agents and Multiagent Systems (MAS) have been offering effective solutions to complex real-world problems. In these solutions, agents solve their local problems, and they communicate with each other to achieve the goal of the system. Moreover, there is no global control over the autonomous agents in these distributed systems. The main benefits of MASs are [57, 78]: *autonomy* (agents are autonomous and responsible for their actions), *scalability* (easier to add new agents to a MAS than adding new functionalities to a monolithic system), *distribution* (MASs can run on different servers in a distributed fashion over the network), *concurrency* (agents in a MAS can work in parallel to solve the tasks), *adaptability* (actions of intelligent agents can be adapted to the environmental changes), and *robustness* (efficient task sharing between the agents allows a MAS to be fault-tolerant to a certain extent). MASs have been used in various applications such as computer games, education, power systems [46], transportation [79], military, logistics [67], bioinformatics, smart grids [66], space, supply chain management, and so on. Two of the important problems in MASs are the coordination of activities of agents and forming agent groups. To tackle the former problem, Foundation for Intelligent Physical Agents (FIPA) developed standard specifications such as an agent communication language, various coordination protocols, and so on. Many agent engineering tools such as Accessible JAVA Agent Nucleus (AJAN), JADE, JASON, and ASTRA have been developed to design MASs and some of them integrate FIPA specifications to solve the coordination problem. The second problem can be solved by using grouping algorithms such as Coalition Structure Generation Problem (CSGP) and clustering solver algorithms, but the MAS platforms and FIPA do not provide the required coordination protocols to coordinate agents into groups.

AJAN has been developed at DFKI as a generic, user-friendly, and modular agent engineering tool. The behaviors of AJAN agents are designed with SPARQL Behavior Trees (SPARQL-BT) which is the combination of Behavior Trees (BT) and SPARQL that is used to access the agent knowledge which is represented as Resource Description Framework (RDF). AJAN does not support Multiagent Coordination (MAC) functionalities and

forming groups of agents. Moreover, it does not provide FIPA coordination protocols. Additionally, there is no SPARQL-BT-based version of those FIPA protocols. Therefore, this work investigates providing SPARQL-BT-based MAC into groups.

1.2 Goals and Approach

The research question of this thesis is **to investigate to what extent a distributed multiagent coordination between (SPARQL) BT-based AJAN agents into groups can be realized**. To answer this question appropriately, the first goal of this thesis is to develop MAJAN as the extension of the agent engineering tool AJAN for the realization and evaluation of multiagent coordination use-cases into groups. For the realization part, this work aims to provide generic coordination BTs which can be used to model the coordination of agents into groups. A web User Interface (UI) is to be provided as part of this work for the evaluation of the results of MAC use-cases. The second goal of this thesis is to implement and evaluate the given MyWELCOME coordination use-cases accordingly with MAJAN.

In order to deliver the aforementioned goals, this work follows a methodological user-centered software development model. The potential users of the system are placed at the core of this model by collecting their feedback and delivering their requirements. This model consists of *Requirements* (where users' requirements are collected through surveys), *Design* (where concepts and UI mockups are designed based on the collected requirements), *Implementation* (where the concepts and UI mockups are implemented), and *Evaluation* (where the potential users evaluate the fully-functional system through surveys) phases, and it is executed once because of time constraints.

1.3 Outline

The remainder of this work is structured as follows:

- **Chapter 2: Background and Related Work**
  This chapter presents the relevant background information, including RDF, SPARQL, MAS, and BTs. Moreover, it describes AJAN, the MyWELCOME application, and its coordination use-cases in more detail. The chapter is concluded with a section that describes the multiagent coordination and the grouping algorithms in other agent engineering tools.
- **Chapter 3: MAJAN: Multiagent Coordination with AJAN**
  Every step of the MAJAN development methodology is detailed in this chapter. Moreover, this chapter provides the results of the steps of the development model.
- **Chapter 4: Language Course Coordination Use-Case**
  This chapter describes the Language Course Coordination (LCC) use-case of the MyWELCOME application and its implementation with MAJAN. Furthermore, the evaluation process and the results of this use-case are provided in this chapter.
- **Chapter 5: Co-Habitation Coordination Use-Case**
  Co-Habitation Coordination (CHC) use-case of the MyWELCOME application, its implementation and evaluation are presented in this chapter.
- **Chapter 6: Conclusion**

Finally, this work is concluded by summarizing the results. Moreover, this chapter presents the limitations and the opportunities for future work.

Chapter 2
Background and Related Work

This chapter presents the necessary background information that is relevant to this thesis. It covers Resource Description Framework (RDF), SPARQL Protocol and RDF Query Language (SPARQL), and Behavior Tree (BT) which are very important for AJAN. Moreover, the MyWELCOME application, its given use-cases, and the coordination in multiagent systems are described in this chapter. The overview of the related work and the comparison of agent engineering tools with respect to their multiagent coordination capabilities concludes this chapter.

2.1 Agent Engineering Tool AJAN

AJAN [1] is an agent engineering tool designed by *A. Antakli et al.* [10] to create and run intelligent agents. It was designed to be domain-independent, modular, extendable, and user-friendly. It provides various features to agent engineers such as modeling SPARQL-BT-based agents, creating and executing them, adding new features via AJAN Plugin System, and so on. In AJAN, agent knowledge is stored as RDF, and therefore, SPARQL queries are necessary to access it. Moreover, AJAN provides a web editor called, AJAN Editor [2], that allows agent engineers to model behaviors of agents with SPARQL-BTs, to create and execute agents in a graphical, user-friendly format.

As stated in section 1.2, the objective of this thesis is to realize Multiagent Coordination (MAC) into groups with AJAN but it does not support such features yet. Therefore, this work extends it with MAJAN to realize and evaluate MAC use-cases. Its architecture and implementation are detailed in the subsections.

2.1.1 Agents and Multiagent Systems

According to [77], *an agent is a computer system that is situated in some environment, and that is able to perform its tasks in an autonomous manner in order to meet its design objectives*. Agents have the following properties [25]:

[1] `https://github.com/aantakli/AJAN-service`
[2] `https://github.com/aantakli/AJAN-editor`

- **Autonomy**: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state.
- **Reactivity**: agents perceive their environment and respond in a timely fashion to changes that occur in them in order to satisfy their objectives.
- **Social ability**: agents interact with other agents to meet their individual or combined goals.
- **Pro-activeness**: agents do not simply act in response to their environment, but they are able to look for alternative solutions for their goals.

Multiagent Systems:
Although single agents have lots of capabilities, there are some complex problems that cannot be solved by a single agent. Solutions for such problems require the capabilities and cooperative work of several agents. A union of agents who work together to meet their individual objectives or to meet the objectives of the union in a decentralized manner is called a **Multiagent System (MAS)**. A MAS provides some advantages as listed in [69].

1. **Efficiency** because of asynchronous parallel computations
2. **Robustness and reliability** since the failure of one or more agents do not crash the system
3. **Flexibility and scalability** since agents can be added and removed
4. **Reusability** because of the distributed nature of the system

2.1.2 RDF and SPARQL

Resource Description Framework (RDF) is a data model to represent and exchange interconnected data on the web, and it is developed by World Wide Web Consortium (W3C). As it is designed to be easily processed by software systems, it is not quickly understandable by humans. Moreover, RDF statements are called *triples* which consist of three components, and they describe the relationships between resources such as people, documents, and data objects. The components of triples are as follows:

1. **Subject** is the *fact* that is described by the triple.
2. **Predicate** represents the *relationship* between the subject and object.
3. **Object** is the *data* that is related to the subject.

These components can be described with Uniform Resource Identifier (URI), blank node, or literal as shown in table 2.1. **URI** is the standard format to identify a resource. A **blank node** is a resource without a URI and it is used as an auxiliary node [42]. A **literal** represents a data value which can be a date, number, or string.

	URI	**Blank Node**	**Literal**
Subject	✓	✓	
Predicate	✓		
Object	✓	✓	✓

Table 2.1: Possible node types of RDF components

Additionally, RDF data can be described as RDF/XML [15], JSON-LD [48] and Turtle [14]. JSON-LD and RDF/XML describe RDF data with JSON and XML syntaxes, respectively. The Turtle family is described as *compact and natural text form* by W3C [3] and it contains Turtle [14], TriG [17], Triples [13], and N-Quads [24] formats.

Table 2.2 describes example RDF triples. In the example triples, the prefixes (table 2.3) are used to improve the readability.

Subject	Predicate	Object
ajan:TestAgent	rdf:type	ajan:Agent
ajan:TestAgent	ajan:agentID	"TestAgent"
ajan:TestAgent	ajan:hasStatus	"Active"

Table 2.2: Example RDF triples

Prefix	Namespace
ajan	http://www.ajan.de/ajan-ns#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#

Table 2.3: Example RDF triples

SPARQL:

SPARQL Protocol and RDF Query Language (SPARQL) [41], published by World Wide Web Consortium (W3C), is a standard query language and protocol for RDF which allows adding, removing, and retrieving data from RDF databases. SPARQL queries include subject-predicate-object triple patterns that are matched with the RDF triples of the database. Moreover, SPARQL supports a wide variety of functionalities such as mathematical operations, if-else statements, utility functions, variable bindings, aggregations, and so on. Furthermore, there are four types of SPARQL queries as listed below:

1. **SELECT** [4] query returns a table including the specified variables and the values that match the patterns in the query.
2. **ASK** [5] returns *true* if there is a match for the given triple pattern.
3. **CONSTRUCT** [6], as the name suggests, returns a set of triples that are constructed based on the given patterns in the **WHERE** clause of the query.
4. **DESCRIBE** [7] returns all the triples that represent all the information found about each resource in the pattern match.

Figure 2.1 describes an example CONSTRUCT query. This query retrieves the agents (line 10) and their IDs (line 11) in WHERE clause. Then, the agents who are not *Active* are filtered out with FILTER EXISTS feature of SPARQL (line 12). Line 13 binds a blank node

[3] `https://www.w3.org/`
[4] `https://www.w3.org/TR/sparql11-query/#select`
[5] `https://www.w3.org/TR/sparql11-query/#ask`
[6] `https://www.w3.org/TR/sparql11-query/#construct`
[7] `https://www.w3.org/TR/sparql11-query/#describe`

```
1. PREFIX ajan: <http://www.ajan.de/ajan-ns\#>
2. PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns\#>
3. PREFIX mac: <http://localhost:8090/rdf4j/repositories/ajan_mac_ontology#>
4.
5. CONSTRUCT {
6.     ?bnode     rdf:type     mac:MACProblemInstance ;
7.                mac:hasParticipants     ?agentId .
8. }
9. WHERE{
10.    ?agentIRI     rdf:type     ajan:Agent ;
11.                  ajan:agentID     ?agentId .
12.    FILTER EXISTS {?agentIRI     ajan:hasStatus     "Active" .}
13.    BIND(BNODE() AS ?bnode)
14. }
```

Figure 2.1: Example SPARQL query

to the *bnode* variable. In the CONSTRUCT clause, the query builds a Multiagent Coordination (MAC) problem (line 6) and attaches only the active agents as the participants (line 7). Finally, lines 1, 2, and 3 represent the prefixes that are used within the query.

Additionally, in SPARQL it is not possible to change RDF triples inside the database as all the available query types are for retrieving data. However, this functionality is provided in SPARQL Update [39] which is a standard language. *INSERT DATA* [8] and *DELETE DATA* [9] are the query types that are provided by SPARQL Update. INSERT DATA is used to add new triples to the database, while DELETE DATA removes the specified triples from the database.

2.1.3 Behavior Trees

According to [27], *a Behavior Tree (BT) is a way to structure the switching between different tasks in an autonomous agent, such as a robot or a virtual entity in a computer game.* BTs have been used broadly in the video game industry, Artificial Intelligence (AI), and Robotics. The characteristics that make BTs so popular are their *modularity*, *flexibility*, *reusability*, *readability*, and *syntactic simplicity*. Moreover, they are very efficient compared to Finite State Machine (FSM) when designing complex behaviors of agents.

BTs are described as directed trees that consist of a *root*, *control flow*, and *execution nodes*. Each node can be defined as a *parent* or *child* node. Except for the root node, all nodes have a parent node. The control flow nodes have one parent and at least one child node. The execution nodes have no children and one parent node. All the nodes have three states, *RUNNING*, *SUCCESS*, and *FAILURE*. A node goes to the *RUNNING* state when it receives an execution signal from its parent. Once the execution is over, the node returns *SUCCESS* or *FAILURE* to its parent if its goal is reached or failed, respectively. The execution of a BT goes from the top (i.e. the root node) to the bottom and from the left to the right. Each parent node sends the execution signal to its child node, receives its status, and acts based on that. According to [27], there are four categories (**Sequence**, **Fallback**, **Parallel**, and **Decorator**) of control flow nodes and two categories (**Action** and **Condition**) of execution nodes. The **Fallback** node is also called the **Priority** node. The definitions of these nodes are given in table 2.4.

[8] https://www.w3.org/TR/2013/REC-sparql11-update-20130321/#insertData
[9] https://www.w3.org/TR/2013/REC-sparql11-update-20130321/#deleteData

Node type	Symbol	Succeeds	Fails	Running
Sequence	→	If all children succeed	If one child fails	If one child returns Running
Parallel	⇒	If ≥ #threshold children succeed	If > #children - #threshold children fail	else
Fallback	?	If one child succeeds	If all children fail	If one child returns Running
Action	<text>	Upon completion	If impossible to complete	During completion
Condition	<text>	If true	If false	Never
Decorator	◊	Custom	Custom	Custom

Table 2.4: The types of control flow and execution nodes [27]

Figure 2.2 describes a simple example BT that consists of one *Priority*, one *Sequence*, two *Action*, and one *Condition* nodes. The execution of this BT starts at the *root node,* which then ticks the priority node. Then the priority node signals its child node from the top (i.e. sequence node) and the sequence node executes the condition node first. If the condition node returns true to its parent node (i.e. sequence node), then the parent executes the action node labeled as *Move forward*. If this action node returns SUCCESS, then the sequence node also returns SUCCESS to its parent (i.e. priority node) which returns SUCCESS to the root node since one of its children succeeded and the BT succeeds. If the sequence node returned FAILURE, then the priority node would execute the action node labeled as *Open the door*. If this action node also failed, then the priority node would return a FAILURE to the root node, which would cause the BT to fail.

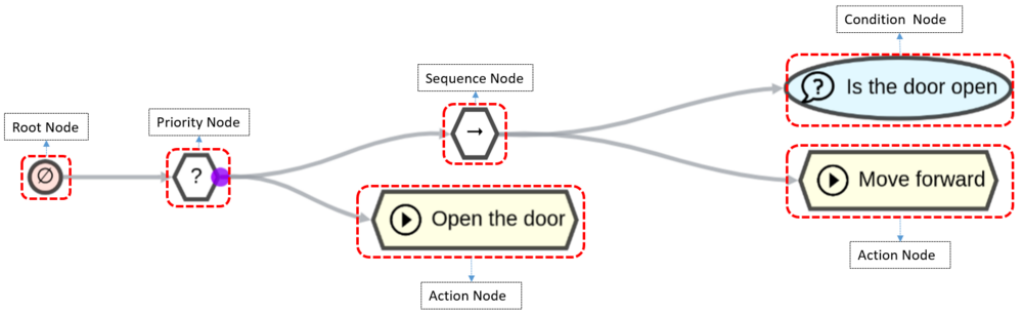

Figure 2.2: Example Behavior Tree

2.1.4 AJAN Architecture

AJAN architecture is described in figure 2.3. It consists of **Execution Service**, **Triplestore**, **Plug-In System**, **Domain**, and **AJAN Editor**.

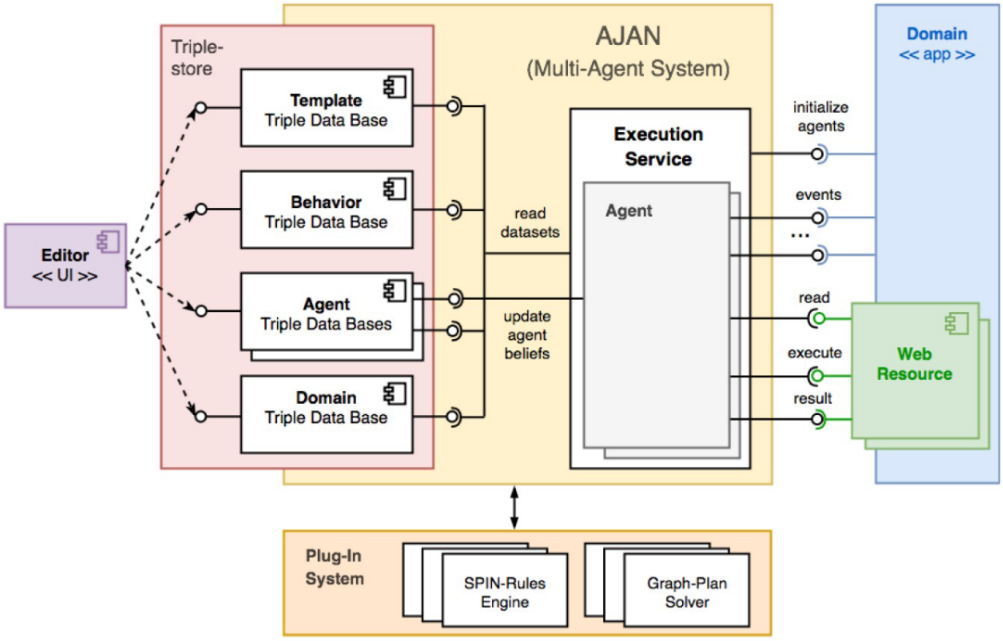

Figure 2.3: AJAN Architecture [10]

The **Triplestore** is a Resource Description Framework for JAVA (RDF4J) database server [64] that maintains four different repositories:

- **Template** repository contains the agent templates to be used to create agents.
- **Behavior** repository contains the behaviors of agents that are SPARQL-BTs.
- **Agent** repository stores the information about the environment and other agents. Each agent has its own specific **Agent** repository to store its local knowledge.
- **Domain** repository stores the domain model with rules, ontologies, and resource descriptions.

Domain in figure 2.3 describes the specific environment/application that AJAN is being used, since it is domain-independent. The interaction between the application and AJAN takes place via HyperText Transfer Protocol (HTTP) requests.

Execution Service [10] is developed with JAVA, and it provides features to manage the life cycle of agents such as *creating*, *deleting*, and *executing* them. For these features, there are Representational State Transfer (REST) endpoints that accept RDF data. In order to create an agent, an HTTP request should be sent to the specific endpoint of the

[10] `https://github.com/aantakli/AJAN-service`

Execution Service component and this request should include information about the agent template, agent name, and initial knowledge such that Execution Service can create the agent with the given name and template and add the given initial knowledge to the local knowledge repository of the agent. The agent templates contain information about events, endpoints, and behaviors of the agent. Moreover, AJAN is event-driven-based, which means behaviors of agents are triggered by events. Such events are created either when receiving a request to the endpoint of an agent or within the BTs of the agent by producing an event.

Plug-In System in AJAN allows extending it with additional functionalities by adding plugins to the system. It keeps AJAN domain-independent such that different functionalities for different domains can be added easily that allowing the core of the AJAN to stay the same for all domains.

AJAN Editor [11] is a web editor that provides functionalities to model the behaviors of agents and manage their life cycle in a user-friendly format. It is developed using Ember JS. An example BT is described in figure 2.4. In order to create a BT in AJAN Editor, users need to select a node from the left side (i.e. Graph Manipulation section), drop it in the middle area, and then connect those nodes with arrows as shown with labels in figure 2.4. AJAN Editor provides lots of SPARQL-BT nodes such as **Action** (#1), **Condition** (#2), **Update** (#3), **Write** (#4), **Sequence** (#5), **Parallel** (#6), and **Priority** (#7) as shown in figure 2.4. The nodes can be customized by making use of the features of the editor such as modifying the SPARQL queries of the nodes, adding a description, and changing the label of the nodes in the **Properties** panel on the right side in figure 2.4.

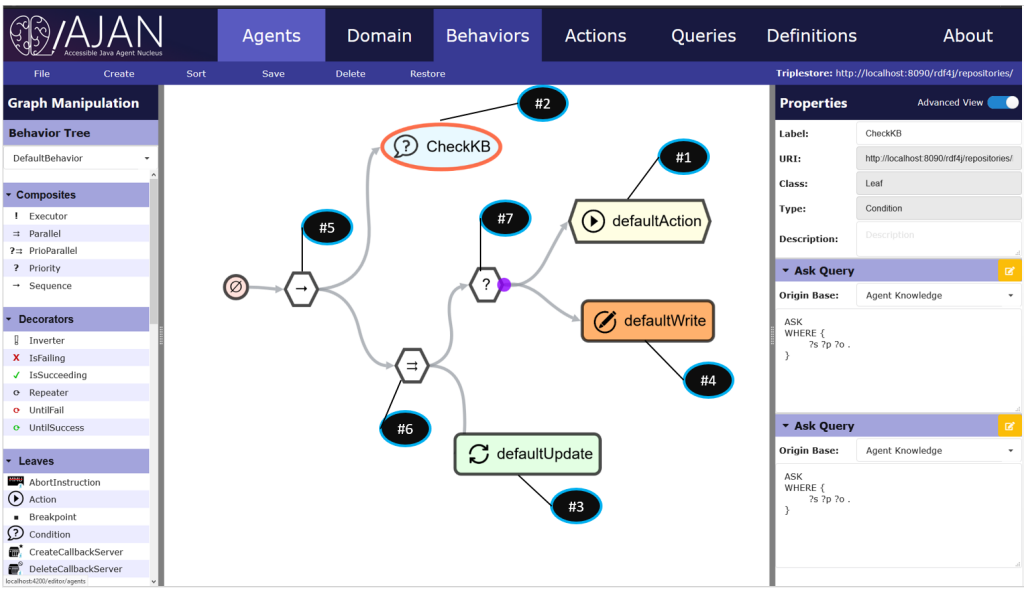

Figure 2.4: An example BT in AJAN Editor

Agents in AJAN follow the Belief-Desire-Intent (BDI) agent architecture. Each of them has endpoints that make the communication available with other agents and external

[11] `https://github.com/aantakli/AJAN-editor`

components. When one of these endpoints receives a message, the message is dispatched as one of the events of the agent. The events of an agent can be connected to specific behaviors (i.e. SPARQL-BTs). So when an event is created, the BT that is connected to it is executed and the RDF message in the event can be accessed via querying within the BT. Furthermore, events can be created by BTs which means that a BT can be triggered within another BT. The BTs make use of two types of data by executing SPARQL queries:

- **Agent Knowledge** is the data that is permanently stored in the Agent repository.
- **Execution Knowledge** is temporarily stored in the Random Access Memory (RAM) which is faster to access and query.

2.1.5 AJAN Implementation

RDF4J [64] is an open-source JAVA framework for working with RDF data. The **Triplestore** is implemented as an RDF4J database server that provides SPARQL endpoints to access RDF repositories. Furthermore, **Execution Service** is implemented as a **Spring-Boot** [75] application, and RDF4J Application Programming Interface (API) is used to connect to the Triplestore. Additionally, the **RDFBeans** framework [8] is used to convert RDF datasets to JAVA beans. To implement the **Plug-In System**, together with RDFBeans, the **PF4J** JAVA framework [59] is used. PF4J makes it possible to keep the Execution Service which is the core of AJAN, simple but extendable. Finally, **AJAN Editor** has been implemented with Node.js [30] and Ember.js [81] frameworks. It provides features to model SPARQL-BTs, create and execute agents, create endpoints and events, and more.

2.2 Mobile Cloud Application MyWELCOME

2.2.1 Mobile Cloud Computing

Mobile Cloud Computing (MCC) is the combination of **Mobile Computing** and **Cloud Computing** fields [32].

- In **Cloud Computing**, cloud providers offer different services such as software, platform and storage to internet applications and users for a cheaper price.
- **Mobile Computing** consists of 3 elements: *hardware*, *software*, and *communication* where they are considered as mobile devices, applications in the mobile devices and mobile networks, respectively.

According to [31], *"**Mobile Cloud Computing** at its simplest, refers to an infrastructure where both the data storage and the data processing happen outside the mobile device. Mobile Cloud applications move the computing power and data storage away from mobile phones and into the cloud, bringing applications and Mobile Computing to not just smartphone users but to a much broader range of mobile subscribers"*. The architecture of MCC is described in figure 2.5.

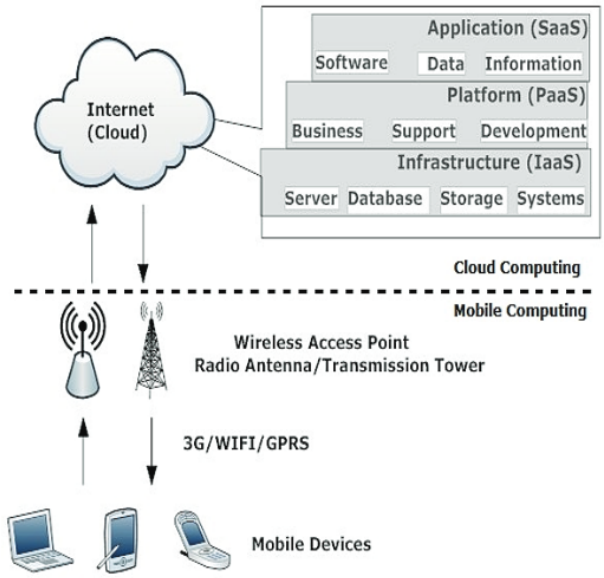

Figure 2.5: Mobile Cloud Computing Architecture. [62]

Moreover, the main benefits of MCC are listed below [40]:

- **More processing power and data storage capacity:** the processing and the storage of the data are offloaded from mobile devices to the cloud servers, which offer much more capacity.
- **Reliability:** cloud servers are more reliable than mobile devices in terms of preventing data loss (by backing up the data) and security issues (by offering services such as virus scanning, authentication, and malicious code detection).

- **Extended battery lifetime:** because of the less computation on the mobile device side, the consumed amount of power is reduced, which extends the battery lifetime.
- **Scalability:** from service providers' point of view, they can extend an application easily and smoothly to meet user expectations.
- **Multitenancy:** service providers can share the resources and costs to provide services to a large number of users.

Nowadays, the amount of Mobile Cloud applications is huge. Some classes of those applications are **Mobile Gaming**, **Mobile Healthcare**, **Mobile Learning**, **Mobile Banking**, and many more.

2.2.2 MyWELCOME Application

MyWELCOME is a mobile cloud-based application that is being developed as part of WELCOME [12] project. The main goal of the MyWELCOME application is to provide services such as registration, language learning, and social and societal inclusion to Third Country Nationals (TCNs) and to authorities to facilitate their decision-making in the reception and integration process of TCNs in Europe. To do so, the aim is to develop intelligent agents that sit in the *cloud* and act as the personal assistants of the TCNs and to provide help, for example, to the language instructors to make more efficient and faster decisions in the language teaching process.

As pointed out on the website of the WELCOME project, its main objectives are listed below [73]:

- Development of technologies to provide to TCNs and authorities to ease the processes of TCN reception and integration
- Development of service coordination and dialogue management technologies for intelligent conversational agents
- Development of multilingual language technologies
- Design of knowledge representation schemata and development of knowledge acquisition and interpretation technologies
- Realization of personalized immersive TCN-agent interaction environments
- Provision of personalized decision support for authorities

There are various situations and legal processes that TCNs encounter and need to go through when they migrate to a country. The WELCOME project aims to implement and provide the so-called *scenarios* to cover situations that TCNs encounter such as registration in the host country, making appointments, the reunification of family members, and language learning. A very important part of this thesis is to implement and evaluate two of these scenarios, Language Course Coordination (LCC) and Co-Habitation Coordination (CHC). Therefore, they are described in more detail in the next section. Briefly, these use-cases require forming groups of TCNs. Since each TCN is represented by an agent, their agents need to coordinate themselves into groups.

[12] `https://welcome-h2020.eu/`

Architecture:

This paragraph explains the responsibilities of the components of the WELCOME system (figure 2.6) that are required for the implementation of the given coordination use-cases.

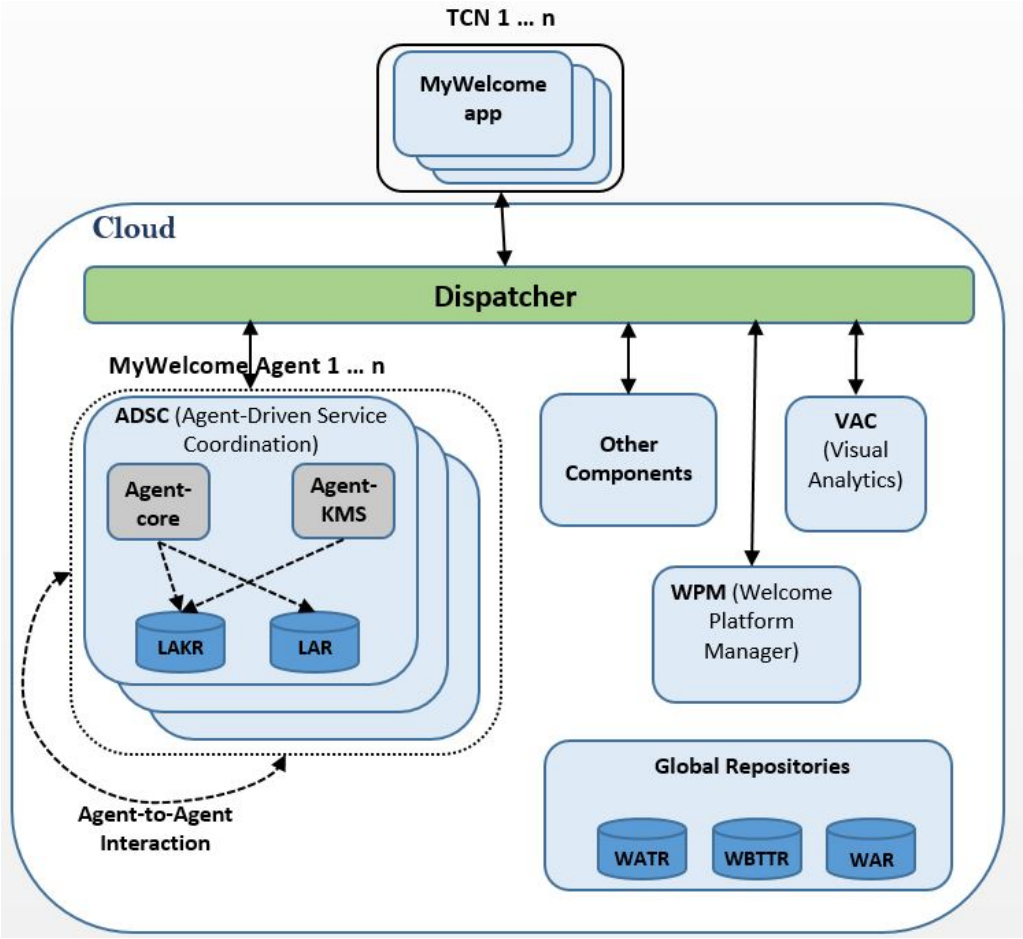

Figure 2.6: WELCOME technical architecture from the perspective of coordination use-cases

- The **Dispatcher** is responsible to mediate all the interactions between all components of the WELCOME system, except agent-to-agent interactions. Agents communicate directly with each other without going through Dispatcher when they are running a coordination use-case.
- Each **MyWELCOME mobile application** provides functionalities to interact with the personalized MyWELCOME agent of the TCN. The TCNs participate in different scenarios of the WELCOME project by performing different tasks in the application. For the given coordination scenarios, the application provides use-case specific features to collect necessary information, pass it to the personalized agent, receive the result of the use-case from the agent and display it in a user-friendly format. All the communication between the MyWELCOME application and compo-

nents of the WELCOME system, including the MyWELCOME agent, goes through Dispatcher.

- **Agent-Core** is AJAN Service (section 2.1), and it is responsible for the creation and execution of MyWELCOME agents in the WELCOME system. Moreover, there is one instance of AJAN/Agent-Core per MyWELCOME agent, and it is the behavioral core of the agent. To create an agent, Welcome Platform Manager (WPM) sends a message to the Agent-Core with the initial agent knowledge. In the given coordination use-cases, MyWELCOME agents need to communicate 1) with each other through their Representational State Transfer (REST) endpoints to exchange data in Resource Description Framework (RDF) and 2) with Visual Analytics Component (VAC) and WPM via Dispatcher to perform use-case related tasks.
- **Agent-KMS** is responsible for maintaining and updating local knowledge (i.e. Local Agent Knowledge Repository (LAKR)) of a MyWELCOME agent. Since LAKR stores the personal information of the TCN and Agent-KMS has access to it, there is one instance of Agent-KMS per MyWELCOME agent. More specifically, Agent-KMS allows the MyWELCOME application, VAC, and WPM to update and query LAKR.
- **Repositories**: There are local and global repositories in the WELCOME system, and they are all RDF4J repositories with endpoints to execute SPARQL queries. Local repositories store the personalized information of TCNs and there is an instance of a local repository per MyWELCOME agent. Global repositories store the generic knowledge for agents, and there is only one instance of each global repository for the whole system.
  - **Welcome Agent Template Repository (WATR)** stores predefined agent templates for the creation of agents by AJAN Service. This repository corresponds to the Template repository in section 2.1.
  - **Welcome Behavior Tree Template Repository (WBTTR)** stores Behavior Tree (BT) templates containing both generic and use-case specific behaviors used by agents. It is the global BT repository for all agents. This repository corresponds to the Behavior repository in section 2.1.
  - **Welcome Agent Repository (WAR)** stores the *ID*, *status*, and *Uniform Resource Locator (URL) address* of all MyWELCOME agents of the WELCOME system.
  - **Local Agent Knowledge Repository (LAKR)** stores the profile information of TCN and other knowledge that is required for the execution of BTs of the agent. It is accessible to Agent-Core and Agent-KMS.
  - **Local Agent Repository (LAR)** stores knowledge about other MyWELCOME agents that are known to this agent. The knowledge contains IDs, statuses, and addresses of other agents. Agents use their LAR to query the addresses of other agents so that they can contact each other during multiagent coordination processes. Moreover, the content of the LAR is populated from the WAR.
- In the context of the given coordination use-cases, the **Visual Analytics Component (VAC)** provides a panel (i.e. Teacher Panel) that serves as the UI for one of the coordination use-cases (namely LCC). MyWELCOME agents receive a signal from

Teacher Panel to start their coordination, and they inform Teacher Panel when the result is ready so that it can be displayed in a user-friendly format.

- **Welcome Platform Manager (WPM)** is responsible for the deployment and management of the components described in figure 2.6. This component creates and kills MyWELCOME agents by sending messages to the AJAN Service. As well as, it handles the population of LAR and the update of the status of an agent in the WAR.
- **Other Components** represent the components that are not essential for the realization of the given coordination use-cases. They are needed for other scenarios of the MyWELCOME application, as there are more than 10 of them. Therefore, other components are not detailed in this thesis.

2.2.3 MyWELCOME Multiagent Coordination Use-Cases

There are two use-cases, **Language Course Coordination (LCC)** and **Co-Habitation Coordination (CHC)**, in the WELCOME project, that require the agents of TCNs to coordinate among each other to form groups based on 1) the individual preferences of TCNs and 2) the system constraints of the use-cases. These use-cases are provided by the user partners, DIFE [13] and PRAKSIS [14], respectively. The role of user partners is to provide scenarios that reflect real-life situations so that the TCNs can solve their problems easily. Moreover, those scenarios are required to be realized in the WELCOME project. The given coordination use-cases should be implemented and evaluated as part of this work. This section provides complete information about the given use-cases.

LCC Description:

Since learning and knowing the **language** of the host country makes the integration process of TCNs easier, host countries provide language courses to them to teach the language quickly. One challenge to meeting this objective is the different learning speeds of students (i.e. TCNs). To resolve this problem, teachers usually divide the students in a classroom into smaller **working groups** manually based on their individual preferences and language progress levels. The objective of the LCC is to automatize this process to support the teacher in her/his decision-making process.

In the LCC use-case, the language course consists of several lessons and there is at least one day in between two lessons. In each lesson, students perform language learning exercises in the classroom. At the end of each lesson, the progress made by the students is individually assessed through means of a specific set of **assessment exercises** for each of these four types of language learning activities: **Reading**, **Writing**, **Listening**, and **Vocabulary**. These exercises are performed by students in their MyWELCOME applications. The grouping of students before each lesson is made by the teacher upon various factors, both **individual preferences**, **system constraints**, and depending on the overall **Course Progress Level (CPL)** of students. Each student is assigned to one working group per lesson. To automize this process, the objective of agents in LCC is to coordinate among each other for finding an (approximately) optimal assignment of students to groups that maximizes the satisfaction of individual and system constraints, and suggest these results to the teacher so that she/he can select one of them. The constraints of this use-case are listed below:

1. **Individual constraints/preferences** need to be set by the students (i.e. TCNs):

[13] https://web.gencat.cat/ca/inici
[14] https://praksis.gr/

(a) **Gender** is a major constraint for the participation of, mostly women but also men, in the language learning course. Therefore, agents of students should respect the preferences of their TCNs about being assigned to a mixed or same-gender group.

(b) **Nationality** is another important constraint, since some students might prefer to be in the same nationality group. Therefore, agents should consider this constraint as well when they form working groups of students.

Each student can select (in the MyWELCOME application) one of the three following options for each preference (i.e. gender and nationality):

(a) **Same:** TCN prefers a group in which the gender of all other group members is the same as his/hers.

(b) **Mixed:** TCN prefers a group in which there are at least one male and one female.

(c) **Don't mind:** TCN does not care about the gender of others in the group.

Moreover, TCNs can specify importance weights for the preferences to let their agents know which preference is more important. The sum of importance weights of the preferences should be one, and each of them should be greater than zero and lower than one.

2. **System constraints** are specified by the teacher:

   (a) Each class has at least **two** and at most **twenty** students.

   (b) Teacher should be able to select the **size of groups dynamically** (e.g. minimum two, maximum five students in each group).

   (c) Each group has students with **similar** course progress levels.

   (d) The teacher should be able to select the **number of students dynamically** who are allowed to be in the same group if they **missed the preceding lesson** (e.g. a group can have a maximum of two students who missed the previous lesson)

Additionally, the assessment of students' progress need to be computed as described below:

- The assessment score of a student is measured at the end of each lesson as the equally-weighted average sum of percentages of correct answers for each assessment exercise type (i.e. Reading, Writing, Listening, Vocabulary).
  - The ***l**anguage **l**earning **a**ssessment **e**xercise score (LLAX-X)* for a given type **X** of assessment exercises **(X in Reading, Writing, Listening, Vocabulary)** denotes the percentage of correct answers given by the student for this type of exercise. An **LLAX-X** score in the interval of **0-49, 50-59, 60-69, 70-79, 80-89, 90-100** is interpreted as *"insufficient"*, *"sufficient"*, *"normal"*, *"good"*, *"remarkable"*, *"outstanding"* progress, respectively.
  - The ***l**anguage **l**earning **a**ssessment score (LLA)* of an individual student is defined as the equally weighted sum of the **LLAX-X** scores he/she obtained for performing four types of assessment exercises.
- The **Course Progress Level (CPL)** score for a student is determined as the average sum of the **LLA** scores for all lessons he/she attended so far.

Since the assessment exercises are performed by students in their MyWELCOME applications, the assessment scores are computed in the MyWELCOME application as well. The MyWELCOME agents are responsible to store the assessment scores in their local knowledge repository and to computing CPL.

CHC Description:

TCNs need to find accommodation for themselves after a certain amount of time in their host country, since they are given accommodation for a limited time when they migrate to the country. However, finding an apartment is a difficult process, and it is even more difficult for TCNs because of various reasons including high rent prices. Therefore, they look for roommates who are similar to them to jointly share apartment costs. In this respect, host countries support them by matching similar TCNs. But this is a challenging task since there are lots of TCNs, they have several individual preferences, and more importantly, the matching is done by employees manually which makes the process very difficult and time-consuming to satisfy the individual preferences of all TCNs. Therefore, the objective of CHC is to automatize this process by creating groups via coordination of agents in which members of the same groups (i.e. TCNs) are most similar to each other. Afterward, agents should inform their TCNs about the contact addresses of other group members of the TCN such that they can contact each other if they want to. In order to form groups that are suitable for all participating TCNs, agents need to consider the constraints of this use-case, as described below:

1. **Individual constraints/preferences** need to be set by TCNs through their MyWELCOME applications:
   (a) **Age:** A TCN could prefer to belong to a group of TCNs with a certain age group(s), as they are more likely to share common interests.
   Selectable preference options: *{Don't mind, 18-25, 26-33, 34-43, 44-50, 51-65, 65-120}*
   (b) **Gender:** A TCN could prefer being a member of a specific gender group.
   Selectable preference options: *{Don't mind, Male, Female, Other}*
   (c) **Family:** A TCN could prefer to belong to a group of certain family type(s).
   Selectable preference options: *{Don't mind, Single Man, Single Woman, Nuclear, Single-parent (mother), Single-parent (father), Extended}*
   (d) **Nationality:** A TCN could prefer being a member of the same or mixed gender group.
   Selectable preference options: *{Don't mind, Same, Mixed}*
   (e) **Religion:** A TCN could prefer to be in a mixed or same religion group.
   Selectable preference options: *{Don't mind, Same, Mixed}*
   (f) **Ethnicity:** A TCN could prefer to be in a mixed or same ethnic group.
   Selectable preference options: *{Don't mind, Same, Mixed}*
   (g) **Apartment preferences:** Following preferences are related to apartment characteristics:
      i. **Location of apartment:** A TCN could specify one or multiple of the following locations as his/her preferred location.
      Selectable preference options: *{Don't mind, Ampelokipoi, Menemeni, Kalamaria, Eleftherio-Kordelio, Evosmos, Agios Pavlos, Neapoli, Pefka, Sykies, Nea*

Efkarpia, Polichni, Stavroupoli, Pylaia, Thessaloniki, Triandria}

ii. **Accessibility for disabled:** A TCN could prefer accessibility service for disabled people in the apartment.
Selectable preference options: *{Don't mind, Yes, No}*

iii. **Rental period:** A TCN could specify a start and end date for his/her preferred rental period.
Selectable preference options: *{Don't mind, Date Range (from-to)}*

iv. **Share with:** A TCN could specify a range, a minimum and maximum amount, as the number of people that he/she prefers to share the apartment with.
Selectable preference options: *{Don't mind, Number Range (from-to)}*

Moreover, TCNs can specify importance weights for the preferences through their MyWELCOME applications to let their agents know which preference is more important. The importance weights should be greater than **zero** and lower than **ten**.

2. **System constraints** of the use-case:
   (a) All cohabitation groups should have the **most similar** TCNs based on their preferences
   (b) All members of a cohabitation group should be able to communicate with each other in at least one **common language.**

2.3 Coordination in Multiagent Systems

In [51], coordination is defined as *"managing dependencies between activities"*. Just like in human societies, it is necessary to have a coordination mechanism in a Multiagent System (MAS) (section 2.1.1) to prevent the chaotic behavior of the total system. Without proper coordination, agents would waste valuable resources and time while trying to either accomplish common goals of the system or maximize their own interests. A few advantages of multiagent coordination are listed below [55]:

1. Since no agent has complete knowledge about the system, and they have their own individual goals, their behaviors might conflict. Thus, a coordination mechanism is required to prevent chaos.
2. Different agents usually have different capabilities in a MAS. In order to solve a large problem together by using their specialized capabilities, they need to coordinate.
3. Coordination improves the overall efficiency of the system since agents avoid performing the same tasks, and exchanging unnecessary information.

There are two types of coordination in a MAS:

1. In a **centralized**, the activities of coordinating agents are controlled centrally. These agents are called coordinated entities. Some works that apply this method are Tucson for Jade [52] and BT extension for Video Games [2].

2. In **distributed**, the agents are not being controlled by any component, and they make decisions and behave autonomously. Some examples of the implementation of distributed coordination are JADE [16], JADEX [20], and ASTRA [28].

Foundation for Intelligent Physical Agents (FIPA) provides the standard specifications for multiagent technology and multiagent coordination to improve the interoperation of heterogeneous agents and the services that they can represent [68]. For the communication of agents, these specifications include FIPA Agent Communication Language (ACL), speech act theory-based communicative acts, content languages, and coordination protocols. This thesis is interested in the **FIPA coordination protocols**, since its objective is to support distributed SPARQL-BT-based multiagent coordination into groups in AJAN. Various protocols such as **Request** [37], **Contract Net** [7], **English Auction** [36], **Blackboard** [76], and **Voting** [60] are some predefined coordination protocols, and they enable agents to synchronize their behaviors by following the steps of the protocols. For example, the **Request protocol** allows an agent to request another agent or agents to do some action and the **Contract Net protocol** is used to share tasks among agents. In this protocol, one agent sends a task to other agents, receives their proposals, and makes selections out of those proposals.

However, none of the predefined multiagent coordination protocols is capable of forming groups of agents. Therefore, the next section discusses several grouping algorithms to build coordination protocols that will enable agents to coordinate themselves into groups.

2.3.1 Classes of Grouping Algorithms

Since it is required to implement the given MyWELCOME coordination use-cases in this thesis and those use-cases require forming groups of agents, the algorithms that are used for grouping purposes and that can potentially solve them are presented. Moreover, it is necessary to design coordination protocols by using these solver algorithms to create groups of agents such that MyWELCOME agents can use and solve the given use-cases.

Some well-known and state-of-the-art grouping algorithms are described below:

1. **Clustering**, also known as **Cluster Analysis**, is the process of dividing a set of objects into clusters based on the similarities between objects. A **cluster** is a **group** that is a collection of objects grouped together because of specific similarities. **Clustering** is a **grouping** that consists of clusters. The objective of clustering is to group objects in a way that objects in the same groups are more similar to each other than objects in other groups.

   (a) **K-Means** [50] is a centroid-based clustering algorithm that is used to divide a set of objects into k groups. Firstly, it randomly selects k objects as the centroids of k clusters. Secondly, it assigns other objects to the clusters based on the distance between the object and the centroid of the cluster. Thirdly, it computes the centroids of the created clusters. Afterward, it iterates through all these steps until either the centroids or members of newly created clusters do not change, or it reaches the maximum amount of repetition. Its worst-case runtime is $O(knT)$, where **k**, **n**, and **T** are the number of clusters, objects, and iterations, respectively.

   K-Means algorithm is **deterministic** when initial centroids are not selected randomly. Moreover, it is **approximate**, and it requires the number of clusters to be specified beforehand. Additionally, it does not support constraints.

(b) **Constrained K-Means Clustering (COP-KMeans)** [72] algorithm is a modified version of K-Means algorithm that additionally considers the instance-level constraints as **Must-link** and **Cannot-link** connections.

- **Must-link** connection/constraint is used when two objects must be in the same group.
- **Cannot-link** connection/constraint is used when two objects must be in different groups.

It tries to satisfy all the given constraints, and it does not produce any solution if any constraint is violated. Moreover, its worst-case runtime is $O(k(n^2)T)$, where **k**, **n**, and **T** are the number of clusters, objects, and iterations, respectively.

COP-KMeans is **deterministic**, **approximate**, it requires the number of clusters to be specified beforehand, it assigns all objects to clusters, and it produces a clustering like the K-Means algorithm.

(c) **Boosted Constrained K-Means Clustering (BCOP-KMeans)** algorithm [56] extends the COP-KMeans to overcome the shortcoming of not producing any solution when a constraint is violated. To do so, it expects the constraints with a priority level, and it iteratively runs the COP-KMeans until the specified iteration number is reached while trying to satisfy the given constraints from the highest to the lowest priority. With this technique, BCOP-KMeans is able to produce a clustering result even when all the given constraints are not satisfied. The runtime of this algorithm is $O(Tt(n^3))$, where **T**, **t**, and **n** are the rounds of boosting process, iterations of COP-KMeans till convergence, and the number of objects, respectively.

BCOP-KMeans has the same characteristics as COP-KMeans but it is capable of producing a clustering result even when some constraints are violated.

(d) **Density-based Spatial Clustering of Applications with Noise (DBSCAN)** [34] is a density-based clustering algorithm in which the clusters are computed based on the density of the objects. Therefore, this algorithm is able to identify clusters with different shapes and sizes. Unlike the algorithms that are introduced above, DBSCAN does not require the number of clusters to be specified beforehand. Instead, it requires two parameters:

- *minPts:* minimum number of objects that are required for a region to be considered as a cluster.
- *epsilon:* maximum distance between two objects to be considered as members of the same cluster

Additionally, the worst-case runtime of this algorithm is $O(nlog(n))$, where **n** is the number of objects.

DBSCAN is **deterministic** and **exact**, and it does not support constraints. Moreover, it produces a clustering result that might contain outliers. Finally, it is sensitive to the *epsilon* parameter.

(e) **Density-based Clustering with Constrained Expansion (CoExDBSCAN)** [33] is an extended version of the DBSCAN algorithm to support the constraints. In addition to the *minPts* and *epsilon* parameters, it requires the list of *constraints* as input. The result of CoExDBSCAN is a set of clusters that satisfies all the

given constraints, but it does not produce any result if any of the constraints is violated. Moreover, its runtime is $O(nmax(T) + nlog(n))$, where **n**, **T**, **max(T)** are the number of objects, set of constraints, and maximum complexity to check the set of constraints, respectively.

CoExDBSCAN is **deterministic** and **exact**, and does not require the number of clusters beforehand like DBSCAN.

(f) **Hierarchical Density-based Clustering (HDBSCAN*)** [23] is a density-based clustering algorithm that builds a hierarchy of clustering results by running DBSCAN iteratively with different *epsilon* values, and then integrates the clusters to produce the most stable clustering result. Therefore, it can compute clusters that have different densities, and it only requires the *minPts* parameter which is easy to select compared to the *epsilon* parameter. Moreover, its runtime is $O(n^2)$, where **n** is the number of objects.

HDBSCAN* is **deterministic**, **exact**, and it does not require the number of clusters beforehand. Besides, it supports constraints unlike DBSCAN, and it produces a solution even if some constraints are violated. Finally, it might produce outliers, meaning that it is not guaranteed to assign all objects to clusters.

(g) **Hierarchical Clustering** [54] algorithm computes a dendrogram that consists of layers of the set of clusters. There are two methods to build the layers:

- **Agglomerative** clustering is a bottom-up technique that starts from the bottom where each object is a single cluster and combines the most similar two objects until all objects are combined into a single cluster.
- **Divisive** clustering is a top-down technique that starts from the top where all objects are in a single cluster and divides the objects into the least similar clusters until each object is a single cluster.

Since a dendrogram is produced as the result of this algorithm, the number of clusters needs to be specified beforehand so that the dendrogram can be cut at the given depth.

Moreover, the runtime of this algorithm is $O(n^3)$. It is **deterministic** and **exact**, but it does not support constraints.

(h) **Adaptive Clustering Ensemble (ACE)** [9] algorithm uses multiple clustering results to ensemble them to compute a single, better clustering result. To do so, it iteratively combines the most similar clusters until there is no further similarity for combination. Lastly, it handles the non-clustered objects by either leaving them as outliers or assigning them to the most similar clusters. In this method, different clustering results can be computed by either using different clustering algorithms or using the same algorithm with different parameters.

It requires the number of clusters to be given beforehand, and its worst-case time complexity is $O(c^2(c + n - k))$, where **c** is the total number of clusters in all clustering results, **n** is the number of objects and **k** is the number of clusters. Additionally, the runtime of the selected clustering algorithms to produce the clustering results is needed to be taken into account. Besides, ACE does not support constraints.

2. **Coalition Structure Generation Problem (CSGP) [63]:** for multiagent systems, focuses on partitioning a given set of agents into mutually disjoint coalitions (i.e. groups) so that the sum of the coalition values (also called social welfare) in the resulting coalition structure (i.e. grouping) is maximized.

   - **Coalition is a group** that consists of agents.
   - **Coalition Structure is a grouping** that consists of coalitions/groups of agents.

   There are several *centralized exact* and *heuristic* CSGP solving algorithms available in the literature, but there is no approach to distributed CSGP solving. Moreover, it is very challenging to solve this problem because of the exponential search space. A centralized CSGP solver algorithm works as follows:

   - **Step 1.** Passing coalitions and their values as input to the algorithm.
   - **Step 2.** The algorithm computes Coalition Structure values by summing coalition values.
   - **Step 3.** The algorithm compares Coalition Structures by their values and computes a ranked list of them (i.e. Coalition Structures) from highest to lowest value.

   (a) **Bi-directional Search Algorithm for Optimal Coalition Structure Generation (BOSS)** [5] algorithm finds an exact solution to the CSGP. To do so, it runs Improved Dynamic Programming (IDP) and Integer Programming (IP) in parallel to find the optimal solution in the search space. Since CSGP is exponential, BOSS is able to find a solution for 27 agents at most. Its runtime complexity is $O(3^n)$.

   BOSS is **deterministic** since it finds the **exact** solution. It does not require a predefined number of coalitions, and it assigns all agents into groups. Moreover, constraints can be considered when computing the values of coalitions. Finally, it produces a rank list of coalition structures, where the first item in the list is the best/exact solution.

   (b) **Agent Code-based Coalition Structure Search (ACS)** [70] is a heuristic CSGP solver algorithm. Unlike BOSS, it is able to run and find a solution for thousands of agents. And this solution is not guaranteed to be an exact solution. Therefore, it is not **deterministic**. Its worst-case runtime is $O(n^3)$. Like BOSS, it returns a rank list of solutions.

3. **Recommender Systems** suggest items (e.g. news to read, movies to watch, music to listen to, products to buy) that would be interesting and preferred to the user(s), based on several criteria. The items are not grouped per user, instead, they are suggested as a list and the same items can be suggested in different lists.

   (a) **Reciprocal Recommender System (RRS)** [58] suggests users to other users. Some usage areas of these algorithms are dating applications to match users and social networks to suggest friends. The challenge in RRS is that both users, who are recommended and who receive the recommendation, should like the matching for the recommendation to be successful. Therefore, it is necessary to compute mutual compatibility between both users by taking into account their individual preferences.

   In order to implement an RRS, the first step is to compute compatibility scores between all users. Since each user would have a different set of in-

dividual preferences, their compatibility scores would be different as well. The second step is to compute the reciprocal scores between pairs of users by using the *Harmonic mean* of compatibility scores [61]. The last step is to compute the list of users for recommendation by sorting the reciprocal scores from the highest to the lowest. Different recommendation lists can contain the same users.

(b) **Group Formation Reciprocal Recommender (Group Formation RRS)** aims to form groups of users, unlike RRS. In [80], there is an idea that investigates whether it is possible to make groups of users from RRS rank lists, but there is no implementation of it yet.

2.3.2 Coordination in Multiagent Engineering Tools

This section presents the well-known agent engineering tools to see to what extent they support the features that are needed to coordinate agents into groups. More specifically, the focus is on the following features:

1. The method to model the **behaviors of agents**, since one of the main goals of this thesis is to design coordination of agents with SPARQL-BTs.
2. Since the aim of this thesis is to design a distributed Multiagent Coordination (MAC), the **centralized** or **distributed** nature of the coordination needs to be checked in the tools.
3. As the coordination protocols are important to develop MAC use-cases, it is necessary to check whether the tools support any **FIPA protocols** or not.
4. As stated in the objectives of this thesis (section 1.2), it is important to have the feature to **coordinate agents into groups** and **evaluate the result** of coordination.

In the following, some well-known agent engineering tools are analyzed based on the four attributes that were mentioned above:

1. **AJAN** (section 2.1) [10] agent engineering tool supports users to design, create and execute SPARQL-BT-based agents, add new features via the plug-in system, and so on. Moreover, knowledge of agents is stored as an RDF and thus, SPARQL queries are necessary to access it. AJAN has been developed with JAVA.

   MAC support attributes:

   (a) Behaviors of AJAN agents are modeled with SPARQL-BTs.

   (b) AJAN does not provide MAC functionalities yet, but its agents act completely based on their individual BTs meaning that they are not controlled by any central component.

   (c) AJAN does not provide any predefined (FIPA) coordination protocol.

   (d) AJAN does not support forming groups of agents and evaluating the grouping result.

2. **JADE** [16] is an open-source software framework implemented with JAVA for the development of multiagent systems. In JADE, there is a middleware that enforces FIPA specifications so that all the systems that are modeled with JADE can be interoperable, which is one of its objectives. Moreover, JADE supports *reactive* or *Belief-Desire-Intent (BDI)* based agents via an extension. The agents in

JADE communicate asynchronously by exchanging FIPA Agent Communication Language (ACL) messages.

MAC support attributes:

(a) JADE provides JAVA classes for different behavior types, and these classes are used to model behaviors of JADE agents. One big disadvantage of JADE is the absence of a Graphic User Interface (GUI) editor to model the behaviors of its agents in a user-friendly format.

(b) JADE agents behave independently based on their individual behaviors. Thus, MAC is distributed in JADE.

(c) Since JADE adheres to FIPA specifications, it provides all the eleven FIPA coordination protocols.

(d) To the best of my knowledge, JADE does not provide any feature or coordination protocol to group agents and evaluate the grouping result accordingly.

3. **JADEX** [20] agent engineering tool was designed on top of JADE by extending its middleware with the JADEX BDI engine such that advantages of both, BDI architecture and middleware, are combined in one platform. JADEX agents are goal-directed instead of task-oriented as in JADE, and they also exchange FIPA ACL messages. JADEX has also been developed with JAVA.

   MAC support attributes:

   (a) JADEX agents are modeled with XML, in which the existing JAVA classes of JADEX are provided. These classes are used for agent modeling.

   (b) JADEX agents act independently like JADE agents.

   (c) JADEX provides six predefined FIPA coordination protocols.

   (d) Like JADE, JADEX does not support coordinating agents into groups and evaluating their results.

4. **JASON** [19] is an agent engineering tool where the agents are modeled with **AgentSpeak** language. This agent-oriented programming language is logic-based, and it supports BDI architecture. JASON interprets the content written in this language to model JASON agents, which communicate by exchanging Knowledge Query and Manipulation Language (KQML) messages. JAVA has been used to implement JASON.

   Multiagent Coordination (MAC) support attributes:

   (a) Behaviors of JASON agents are modeled with AgentSpeak language.

   (b) MAC in JASON is distributed since agents' behaviors are not controlled by a central component.

   (c) JASON does not provide any FIPA protocol originally, but they can be implemented via an extension according to [6].

   (d) JASON does not provide the features to group agents.

5. **J-Behavior Trees** [18] extends JADE to support modeling agent behaviors with BTs since JADE is not BT-based. To do so, this work provides several JAVA classes for different BT tasks such as *Sequence*, *Parallel*, *Action*, and *Condition*. This extension

has been written in JAVA, like JADE, and it also does not provide any editor to model agent behaviors in a user-friendly format.

MAC support attributes:

(a) This extension allows modeling behaviors in JADE with BTs. However, those BTs are not event-driven and SPARQL-based, which are two very important advantages of AJAN (section 2.1).

(b) MAC is distributed, since this is an extension of JADE.

(c) This work provides one FIPA coordination protocol as a proof of concept.

(d) This work does not provide any functionality for the grouping of agents.

6. **BOCHICA** [74] is a domain-independent agent engineering framework that is used to build multiagent systems. In BOCHICA, DSML4MAS is used as the modeling language to model MAS models, which are converted to platform-specific models and executed afterward. JADE, JADEX, and JACK can be used as platforms. Since BOCHICA is generic, it can be extended with additional concepts and coordination protocols.

   MAC support attributes:

   (a) Agents and their behaviors are modeled with DSML4MAS.

   (b) MAC is distributed since agents are not controlled centrally.

   (c) DSML4MAS supports a few FIPA coordination protocols.

   (d) BOCHICA does not provide features to form groups of agents.

7. **ASTRA** [28] is an agent programming language that was designed based on AgentSpeak and JAVA and provides lots of features on top of their basic functionalities. Additionally, ASTRA provides an Eclipse plugin to easily execute an ASTRA code. Moreover, agents exchange FIPA ACL messages to interact with each other. Finally, ASTRA has been written in JAVA.

   MAC support attributes:

   (a) Behaviors of ASTRA agents are modeled with ASTRA agent programming language.

   (b) Behaviors of ASTRA agents are not controlled by any component and agents decide autonomously.

   (c) ASTRA provides six FIPA coordination protocols.

   (d) ASTRA does not provide features to create groups of agents and evaluate them.

8. **Tucson for JADE (Tucson4Jade)** [52] is a JAVA library that is integrated into JADE middleware and provides coordination functionalities of Tucson as interfaces to JADE agents. Tucson framework provides coordination as a service to agents via tuple centers, such that behaviors of agents are managed centrally to achieve overall successful coordination. In Tucson, agents are coordinated entities. Tucson4Jade allows agents to coordinate centrally as in Tucson or distributed as in JADE. Moreover, it has been developed in JAVA.

   MAC support attributes:

(a) Agent behaviors are designed in the same way (i.e. with behavior classes) as in JADE since this work extends it.

(b) Agents are controlled by Tucson middleware in Tucson4Jade and therefore, the coordination is centralized.

(c) No FIPA coordination protocol is predefined in this work.

(d) Tucson4Jade does not coordinate agents into groups.

9. **Behavior Tree Extension for Video Games** [2] extends the capabilities of behavior modeling in the video game industry such that non-player characters (i.e. agents) can coordinate with each other to present non-artificial and more intelligent actions to users. Event-driven behavior tree-based modeling is widely used in the video game industry.

   MAC support attributes:

   (a) Behaviors of agents are designed with event-driven BTs.

   (b) The coordination is centralized since the behaviors of agents are managed by a single and central BT. Moreover, the agents must share the same memory space in this work.

   (c) This work provides one predefined FIPA coordination protocol.

   (d) This work does not support any functionality to form groups of agents.

2.3.3 Summary

The goal of this thesis is to realize and evaluate a (SPARQL) Behavior Tree-based distributed Multiagent Coordination into groups as stated in section 1.2. To meet this goal, this work develops MAJAN and this section compares it to other agent engineering tools that are described with regard to their capabilities of supporting distributed (SPARQL) BT-based MAC, in section 2.3.2. Table 2.7 describes the comparison to show why MAJAN is necessary to be developed.

Tools	FIPA Protocols	Coordinate Agents into Groups	Behavior-Tree based	Other
JADE (Bergenti et.al, 2005)	Provides 11	No	No	-
JADEX (Braubach, 2005)	Provides 6	No	No	-
JASON (Bordini et. al, 2007)	Provides 1 via extension (Vivancos et. al, 2011)	No	No	Central coordination via central Conversation Manager component
J-Behavior Trees (Bojic et. al, 2011)	Provides 1	No	Non event-driven and Non SPARQL BT based	-
BOCHICA (Warwas et.al, 2012)	Does not provide	No	No	FIPA Protocols can be added via extension
ASTRA (Collier et.al, 2015)	Provides 6	No	No	-
Tucson for JADE (Omicini et al, 2015)	Does not provide	No	No	Central coordination via Tuple Center
BT Extension for Video Games (Agis et al, 2020)	Provides 1 via communication nodes	No	Yes	Central BT to control agents Central memory space shared among coordinating agents
AJAN (Antakli et.al, 2019)	Does not provide	No	Yes (SPARQL-BT based)	-
MAJAN	Provides **(SPARQL-BT based)** initially 3 **coordination protocols**	Provides generic features to **coordinate** agents **into groups** as **Coalition Structure Generation** and **Clustering** problems	Yes **(SPARQL-BT based)**	Provides feature to **Evaluate** the grouping results of **multiagent coordination** Provides feature to **Monitor** the actions of **coordinating agents**

Figure 2.7: MAJAN comparison table

The capabilities that MAJAN provides compared to other tools are listed below:

1. As shown in table 2.7, none of the tools is SPARQL-BT-based. Only *J-Behavior Trees* and *BT Extension for Video Games* are BT-based and in the former, BTs are not event-driven, which is an important aspect of the BT concept.
2. Although *BT Extension for Video Games* is BT-based, it supports only the **centralized**

coordination of agents. Meaning that agents in the coordination are controlled by a single BT and they have to share the same memory space.

3. None of the tools provide any features to form groups of agents. However, MAJAN provides SPARQL-BT-based coordination protocols to coordinate agents into groups.
4. Since tools do not provide a grouping feature, they also do not provide any GUI to evaluate the grouping results. However, MAJAN allows users to visualize and evaluate a grouping result of a MAC process by comparing it to a central grouping algorithm result.
5. Moreover, FIPA coordination protocols are useful for MAC but their SPARQL-BT-based versions have not been developed yet. This work develops and provides one FIPA and two grouping coordination protocols with SPARQL-BTs as part of MAJAN.

Another goal of this thesis is to implement the given MyWELCOME use-cases (section 2.2.3) which cannot be solved with other tools since they do not support forming agent groups through coordination. For this and other reasons that are listed above, this thesis develops MAJAN. Its development process is described in section 3.

Chapter 3
MAJAN: Multiagent Coordination with AJAN

This chapter presents the methodological development of MAJAN [15] and the details of each phase of the development process. The motivation for developing MAJAN is given in section 2.3.3 by comparing it to other well-known agent engineering tools (section 2.3.2).

3.1 Development Methodology

The subsequent sections present the methodology applied for the development of MAJAN and describe the details of the selected life cycle model.

3.1.1 Overview

The development of a software product might take a long time for various reasons. Thus, **life cycle models** are used to abstract the steps of the development process. Lots of life cycle models such as the well-known **Waterfall** (figure 3.1) and **Star** (figure 3.2) models have been developed so far.

[15] `https://github.com/AkbarKazimov/MAJAN`

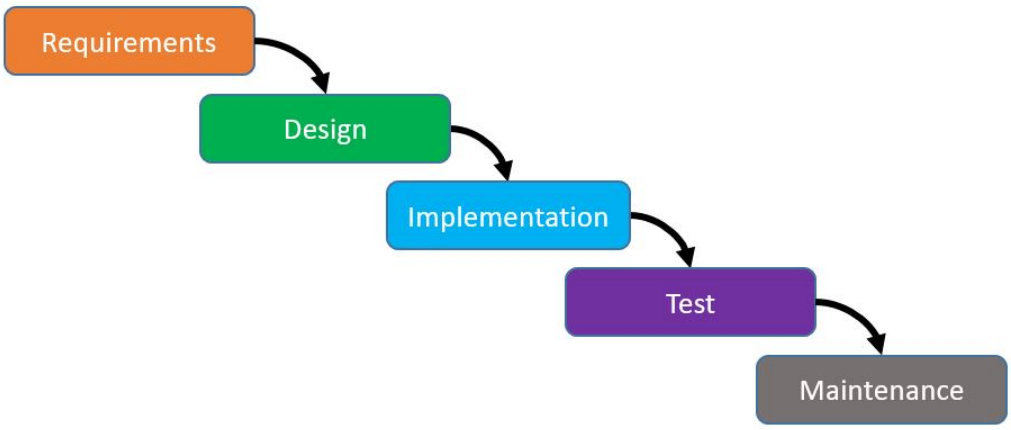

Figure 3.1: Waterfall Model [49]

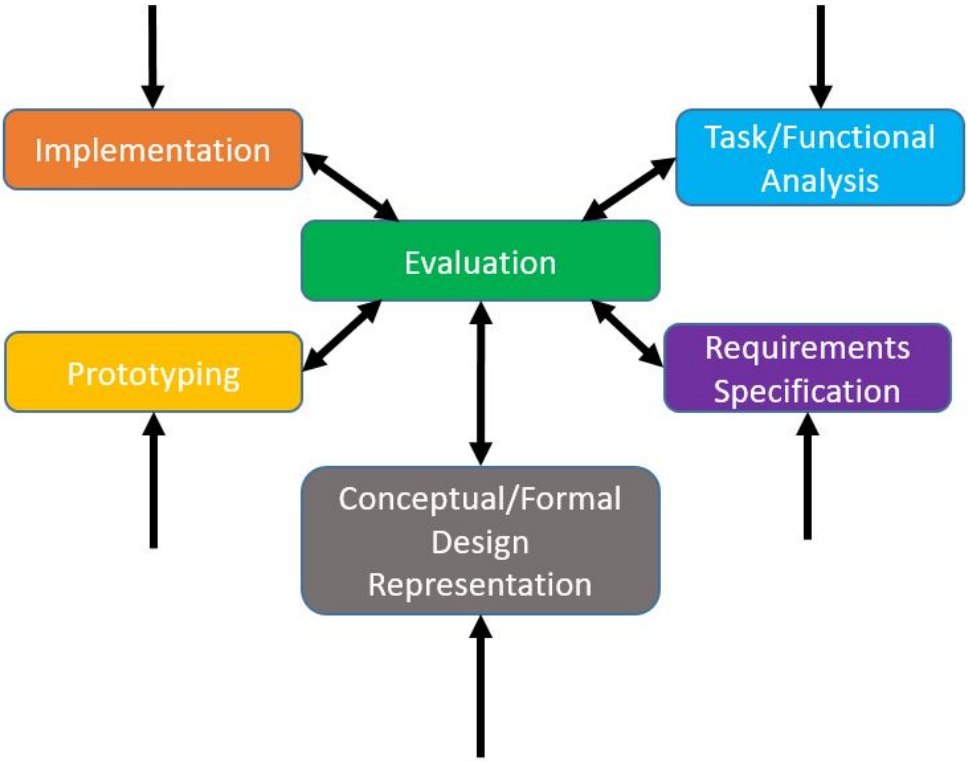

Figure 3.2: Star model [49]

- **Waterfall** model, as shown in figure 3.1, is the first Software Development Life Cycle (SDLC) model, and it enforces completing each step before proceeding to the next one [65].

- **Star** model (figure 3.2) does not enforce following the steps in any particular order. However, it is necessary to complete the **Evaluation** phase before moving from one step to any other [65].

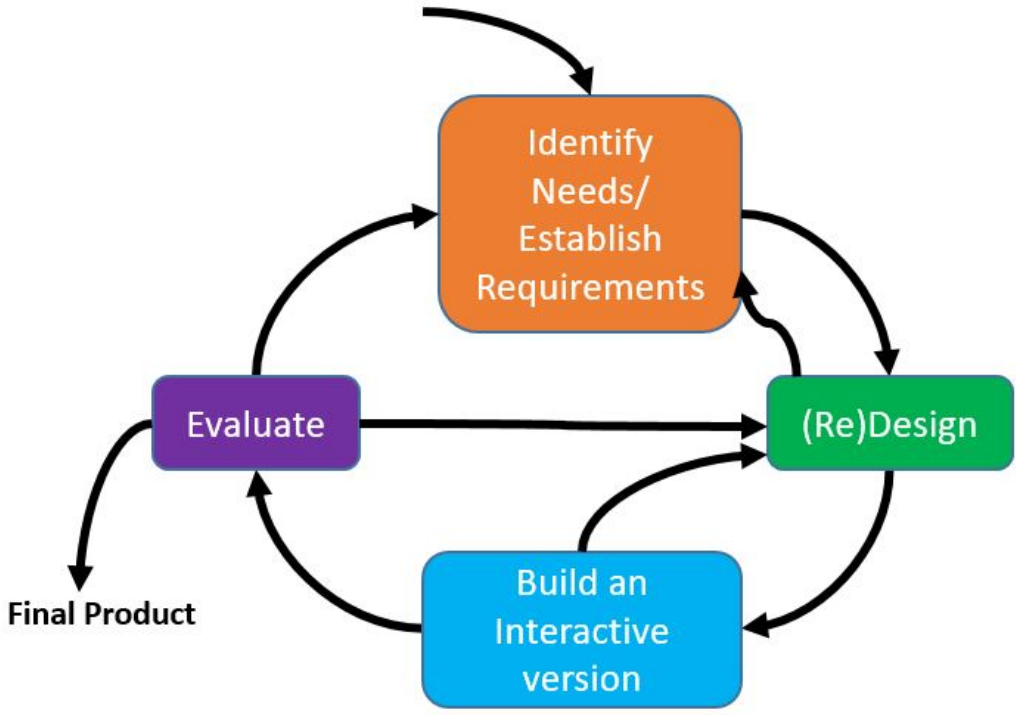

Figure 3.3: The life cycle model suggested by [49]

The model which was suggested by [49] is illustrated in figure 3.3 and this work used an adapted version of it. The main idea of this model is to start by **Establishing Requirements** and continue with **Designing** single or multiple alternatives before proceeding to the next steps. Then the whole process needs to be completed with an **Evaluation** step which produces the final product that satisfies the requirements. This model was modified in a way that its steps were ordered as a cycle (figure 3.4). Furthermore, one iteration of the model in figure 3.4 was applied because of the limited time of this project. In this model, the users are at the core of the development, since the first step is to collect their requirements.

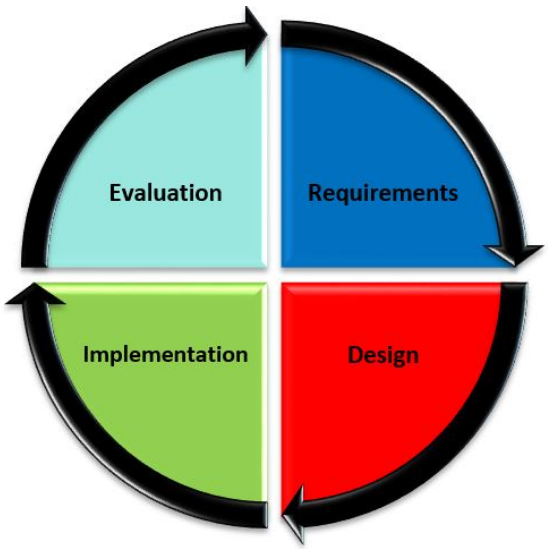

Figure 3.4: The selected life cycle model for the development of MAJAN [49]

3.1.2 Requirements Phase

Personas:

In order to collect the requirements from users, **Expert**, **Non-expert**, and **End-user** personas were created. They helped to separate the users and clearly identify the classes of requirements that should be collected.

1. **Experts** were AJAN developers who know how to use and programmatically extend AJAN Service and AJAN Editor.
2. **Non-experts** were AJAN agent engineers who were using AJAN Editor to model and run AJAN agents by mostly working on the level of SPARQL-BTs.
3. **End-users** were non-AJAN users who saw the system as a black box. They provided use-cases to be developed by non-expert or expert users, and they were only concerned with the results of the given use-cases.

Moreover, expert and non-expert users were expected to be familiar with Multiagent Coordination (MAC) concept since MAJAN provides features to design MAC use-cases.

Requirements Elicitation:

Initially, some basic requirements were derived from the objectives (section 1.2) of this thesis. Then, initial concepts and UI mockups of MAJAN were designed based on these initial requirements. Afterward, a questionnaire (that is provided in appendix A) was prepared by using Google Forms, to collect the requirements of potential users.

Here, a summary of the questionnaire is provided. It was divided into the following sections:

- **User Context** (section A.1): to collect users' background knowledge about MAC and the technologies that were needed to use MAJAN.
- **MAC Use-Case Realization** (section A.2): to collect users' requirements about the suggested concepts to model agent behaviors, create agents, and execute them in a

MAC use-case.

- **MAC Use-Case Monitoring and Evaluation** (section A.3): to collect users' requirements about the suggested concepts to monitor the activities of coordinating agents and to comparatively evaluate the grouping result of a MAC use-case with a centrally running grouping algorithm.
- **MAJAN Guideline** (section A.4): to collect user requirements about 1) the complete workflow, starting from the integration of grouping algorithms into AJAN to the evaluation of grouping solutions, and 2) the overall structure of the guideline document.

The survey was conducted online with three users, since only these people were technically proficient with the concepts which were aimed to be achieved in MAJAN. Before the scheduled slot, the signed consent forms were collected from all participants for data privacy. Then, users answered the questions and completed the survey on their laptops or Personal Computers (PC). During this time, I was online and available in case they had a question or a problem. Once they completed the online survey, a recorded interview via Skype was taken place to ensure that everything described in the survey was clear to them and also to collect their additional feedback, whether it was about something that was forgotten to ask or they forgot to answer in the survey.

3.1.3 Design Phase

In this step, the initial concepts were updated based on the requirements that were collected from the participants in the **Requirements** phase. The goal was to collect as many as requirements from users, but all of them were not feasible for implementation because of time constraints. Therefore, the requirements were prioritized, and the UI mockups were updated accordingly.

3.1.4 Implementation Phase

The first goal in this phase was to implement MAJAN concepts and mockups which were designed in the previous phase. By the end of this step, the necessary UI elements and the back-end functionality were ready in order to be able to evaluate the system as a whole. After the implementation of the prioritized functionalities, the given coordination use-cases of the MyWELCOME application were implemented to meet the requirements of end-users. More implementation details of Language Course Coordination (LCC) and Co-Habitation Coordination (CHC) use-cases are given in sections 4.3 and 5.3, respectively.

3.1.5 Evaluation Phase

In this phase, a survey (in appendix B) was created using Google Forms to evaluate the usability of MAJAN to improve it based on users' feedback. Moreover, the survey aimed to identify the additional desired functionalities. Besides, the satisfaction of users was measured to estimate the success of this work. Since the MAJAN Guideline document was provided to users so that they can be able to use the features of MAJAN, this survey also measured its usability of it.

The survey started with a verbal introduction, where the main features of MAJAN and the objective of this survey were introduced. Then the participants proceeded to the first of four sections as described below:

1. **User Background** (section B.1) questions collected the background knowledge of participants about their familiarity with AJAN, SPARQL-BTs, grouping algorithms, and their field of expertise.
2. **Tasks** (section B.2) were the assignments to be completed by the participants by interacting and using the functionalities of MAJAN. These tasks described the typical workflow of developing a multiagent coordination use-case (into groups) with MAJAN.
3. **System Usability Scale (SUS) [22, 45]** (section B.3) questions were asked after the participant completed the tasks. SUS was selected since it is an industry standard to quickly measure the usability of a system. Moreover, it contains ten questions that should be answered on a scale from **one** (*strongly disagree*) to **five** (*strongly agree*) after the participant interacts with the system (appendix B).
4. **Complementary** (section B.4) questions aimed to collect additional feedback about MAJAN. To do so, these questions cover the points that were not covered by SUS questions.

The survey was complete after the participant filled out all four sections. During the survey, I observed the interactions of participants with the system to gather more information, including the tasks that participants struggled to complete and the UI elements that were not fully understood at the first glance. Because of the Corona regulations, the survey was conducted online. To ensure that all the participants received the same survey and participated in the same environment so that there was no bias, I set up the system and the survey on a laptop and allowed the participants to complete it on this laptop by joining with the remote control software, TeamViewer.

Although the survey did not include any personal questions, the session was not recorded and the answers were collected anonymously, a signed consent form was collected from each participant since the interviewer knew the identities of the participants, and the answers could be matched with the respective participants by the interviewer.

3.2 Development Results

This section presents the results of each phase of the life cycle model that was executed. More specifically, the initial concepts, requirements collected from users, mockup designs, implementation, and evaluation of the whole system are presented without going into too much detail.

3.2.1 Initial Concepts

Three main preliminary concepts were derived from the thesis objectives (section 1.2) to suggest them to the users as the initial solutions of this work to collect their requirements, either to improve the concepts or to design completely new ones. The initial concepts are given below:

- **Template coordination BTs** to develop multiagent coordination use-cases into groups.
- **Monitoring the activities** of coordinating agents since AJAN did not provide such functionality yet.
- Functionalities to **view** the result of multiagent coordination, **execute** a centrally running grouping algorithm and **compare** these results accordingly.

The paragraphs below describe the UI mockups for the initial concepts.

Template Behavior Trees:
In order to realize a MAC use-case successfully, it is necessary to design the behaviors of coordinating agents in a way that they can coordinate their activities to achieve a common goal of the Multiagent System (MAS). To do so, a coordination protocol should be created, and it should be executed by the coordinating agents. FIPA provides several predefined protocols as the standard for agent coordination. Since there is no implementation of such protocols with SPARQL-BTs, the suggestion was to provide template BTs for the following coordination protocols:

1. **FIPA Request protocol** [37] (section 2.3) allows an agent to request another agent or agents to do some action. This is the base protocol of all FIPA protocols since it contains the basic steps for successful agent communication. In this protocol, agents exchange necessary information with each other, unlike Coalition Structure Generation Problem (CSGP) and Clustering coordination protocols in which agent groups are formed.
2. **CSGP Coordination protocol** was proposed to be developed by this work to solve CSGP to create groups of agents in a multiagent system. This protocol should be generic such that different solvers such as BOSS [5] and ACS [70] can be used as a plugin within the protocol without needing to modify anything else. CSGP and its solver algorithms were introduced in section 2.3.1.
3. **Clustering Coordination protocol** was suggested to be developed to solve the Clustering problem to create clusters of agents through multiagent coordination. In this protocol, it should be possible to use different clustering solvers, such as HDBSCAN* [23] and DBSCAN [34]. The clustering problem and its solver algorithms were discussed in section 2.3.1.

By using these coordination protocol BTs, AJAN users can coordinate agents into groups. The main advantage of coordination BTs is their reusability, such that users do

not need to create every time manually from scratch. Additionally, these BTs should be generic and use-case independent so that they can be customized for different use-cases. Moreover, it was suggested to provide generic coordination BT nodes by extending AJAN appropriately such that the provided nodes can be used not only for the given MyWELCOME application use-cases but for any use-case to coordinate agents into groups.

Monitoring The Activities of Coordinating Agents:
Since a multiagent system might consist of hundreds of agents, it is very useful to be able to monitor their activities in a user-friendly way. Although it is possible to view the logs of agents in NetBeans, which is an AJAN feature, it was not designed for multiple agents. Since agents do not attach their names to the logs that they produce in NetBeans, it is very difficult to identify the producers of logs. Moreover, all agents log all of their activities in NetBeans, which makes it very difficult for debugging. Therefore, the objective was to provide functionalities to monitor the specified activities of coordinating agents. This functionality is described in figure 3.5 as the initial UI mockup. The **Multiagent Coordination Logs** panel in figure 3.5, displayed the logs of activities of coordinating agents. Each logging information should consist of *time*, *agent name*, *use-case title*, *name of an algorithm* if it exists, *activity*, and *activity status*. Activity status could be *Success, Failure, or Running*. The *Activity* could be specified flexibly by users based on their needs in the use-case they run.

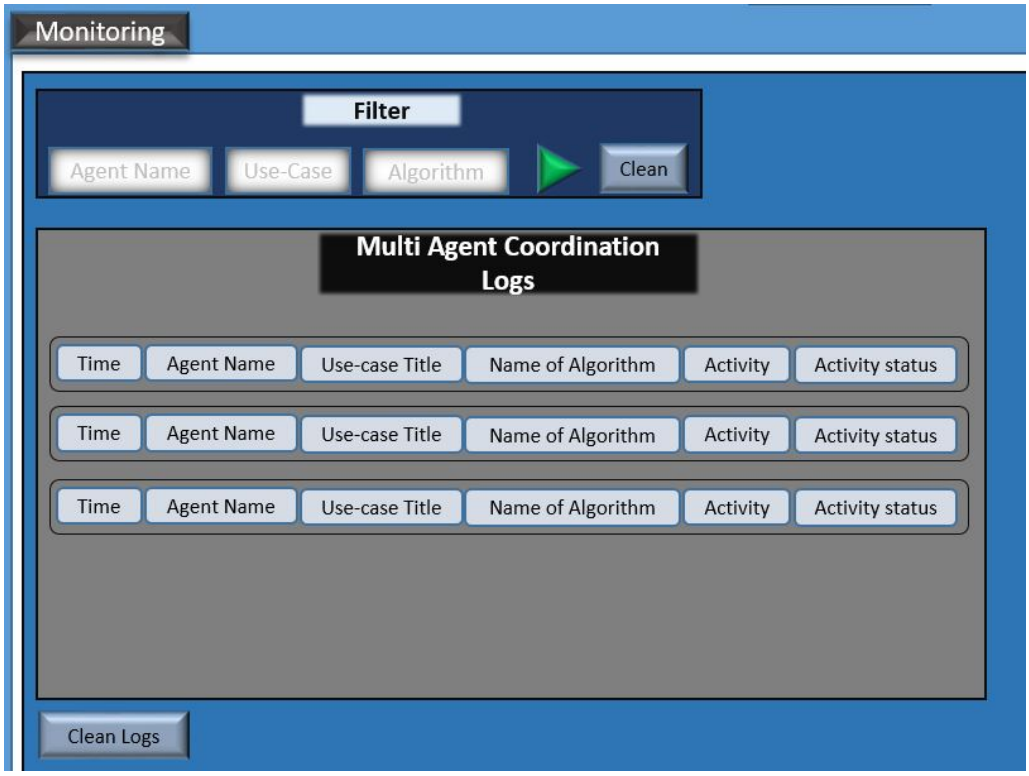

Figure 3.5: UI mockup of the Monitoring functionality

Evaluating Multiagent Coordination Result:

In order to evaluate the correctness of a grouping result of a MAC process, the concept was to provide the features for the visualization and comparison of the grouping results of a MAC process and a centrally running grouping algorithm. The initial mockup describing the evaluation functionalities is shown in figure 3.6. In the **MAC Solution** section, users needed to specify the RDF repository that stored the grouping result of a MAC process. Then the system would retrieve the results by sending the necessary SPARQL query to the specified repository. Once the results were fetched, the system would parse and display them in a user-friendly format rather than RDF data. Since there could be multiple results in one repository and each result could have a single or a set of solutions, there were two lists in figure 3.6 to display the results and their solutions, respectively. Additionally, more detailed information such as *use-case title*, *solver algorithm*, *runtime*, and *number of agents* for each result would be shown if they existed in the specified repository.

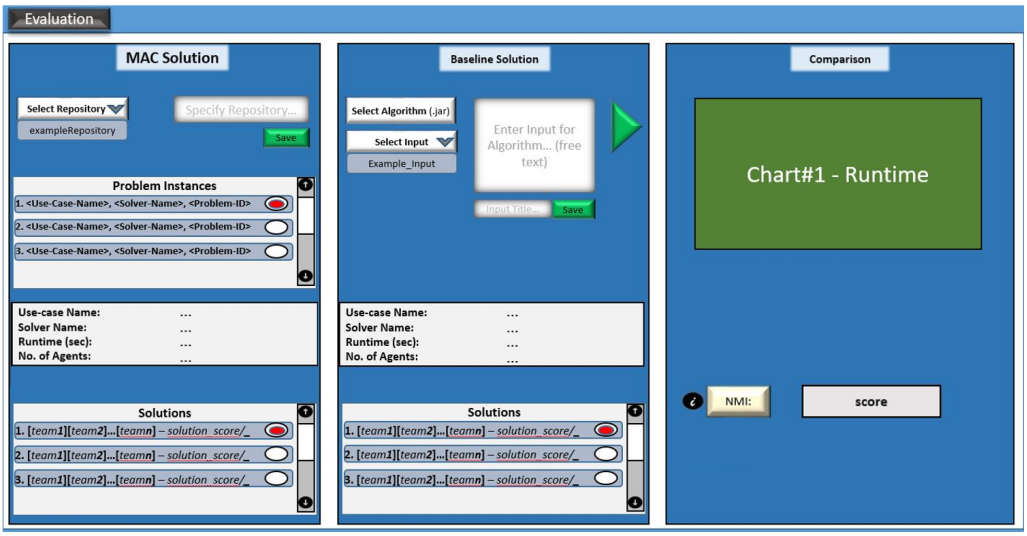

Figure 3.6: UI mockup of the Evaluation functionality

In the **Central Solution** section, users would upload centrally running grouping algorithms as *.jar* files, select one of them, pass corresponding input to it and run. The system would execute the specified algorithm, retrieve its output, parse the output and display it accordingly in the **Solutions panel** of the **Central Solution** section. Besides, the system would display more information about the execution as shown in figure 3.6.

In order to comparatively evaluate the solutions, users needed to select one solution from **MAC Solution** and one from **Central Solution** sections according to this suggested concept. Afterward, the system would display 1) the runtime information in a chart and 2) the Normalized Mutual Information (NMI) score [35] of the selected solutions as shown in figure 3.6. NMI is a standard metric to measure the similarity of two grouping results.

3.2.2 Elicited Requirements

This section summarizes the requirements that were collected via the online survey (section 3.1.2). The questions of the survey are provided in appendix A.

User Background:

Firstly, the background knowledge of users about AJAN, RDF, SPARQL BTs, and JAVA was collected via questions 1-9 in section A.1 since the expert and non-expert users were expected to be familiar with these technologies. Based on the answers, they were very familiar (4 out of 5 on average) with the aforementioned technologies.

Secondly, it was identified that all participants had already used AJAN and AJAN Editor in previous projects, but two of them never extended AJAN Editor (questions 10-12 in section A.1). As well as, one of the users never created a SPARQL BT node and never used AJAN Plugin System before (questions 13-14 in section A.1).

Moreover, all users were familiar with the MAC concept and FIPA coordination protocols (questions 14-19 in section A.1). Furthermore, I extracted that only one of the users was familiar with the algorithms to form groups of agents (questions 20-22 in section A.1).

Therefore, the necessary instructions and references about these less-known topics should be provided in the MAJAN Guideline document.

Multiagent Coordination use-case realization:

This section asked questions about realizing a MAC use-case. The first step of developing a MAC use-case is to design the behaviors of agents via SPARQL BTs in AJAN Editor (concept 1 in section A.2). In this regard, users strongly preferred to have the use-case independent coordination nodes such that the use-case related computations can be done in separate nodes with SPARQL queries.

Moreover, they approved the suggestion of having generic BT templates (concept 4 in section A.2) that can be used when developing MAC use-cases without needing to design every time from scratch.

To be able to use the grouping algorithms (section 2.3.1) to form groups of agents in AJAN, users confirmed that AJAN Plugin System was the preferred way to integrate such algorithms (concept 2 in section A.2). Alternatively, one of the users suggested that another solution could be providing the algorithms as a web service and calling them via Web interfaces but using AJAN Plugin System was more efficient as it eliminates the network round-trip. Additionally, users wanted to have specific BT nodes for each grouping algorithm such that the purpose of the node can be understood by its name without needing to look into the SPARQL queries within it.

Another very important aspect of MAC is the communication between agents of MAS. At the time of conducting the requirements survey, AJAN (section 2.1) provided *Action* and *Message* BT nodes which supported the communication between an agent to another agent and an agent to a software component. Participants stated that they do not want any modifications to be made to the communication technique in AJAN (concept 3 in section A.2). Moreover, they wanted to have at least one FIPA coordination protocol to understand the communication and interaction among agents.

Participants did not specify any requirements for the creation (concept 5 in section A.2) and the execution (concept 6 in section A.2) of agents, since AJAN Editor already provided such functionalities.

Monitoring and Evaluation of Multiagent Coordination:

In this section of the survey, the UI mockups of **Monitoring** (figure 3.5) and **Evaluation**

(figure 3.6) features were presented to collect the requirements for the proposed features. Below, the requirements of each panel in MAJAN are described:

- Participants liked the suggested **Monitoring Coordination** functionality, and they stated that they would need to use the feature practically to provide additional requirements.
- There was not any major requirement about the **MAC Solution** panel in the **Evaluation** section. One participant required a clarification about the *"Specify Repository"* field therefore, this title was modified to be self-explanatory, and also, the necessary explanation was provided in the MAJAN Guideline document.
- Overall, participants liked the **Baseline Solution** panel, but one of them suggested an alternative approach where the result of the centrally running grouping algorithm would be described in RDF. Since this alternative approach did not affect the overall solution provided by MAJAN, it was given a low priority so that it would be implemented after the implementation of main functionalities if there is enough time left. Besides, participants preferred to have some information about the selected grouping algorithm in the UI.
- In the **Comparison panel**, participants preferred to have a Bar chart to illustrate the runtime information. Moreover, one of them mentioned that it would be nice to be able to save the chart as a picture. Furthermore, they required to display the names of the selected solutions in this panel for better visualization.

Additionally, one of the participants preferred to have the functionalities to create agents and execute them in MAJAN in addition to AJAN Editor. This was a nice-to-have requirement since the requested features already exist in AJAN Editor.

Guideline document:

In this section, the complete workflow of using MAJAN to develop and evaluate a MAC use-case was given, and it was proposed to develop the MAJAN Guideline document with the same workflow. Additionally, the guideline document should include comprehensive instructions (with screenshots) for every single step of the workflow. Participants did not provide any further requirements in this section.

The requirements surveys provided valuable feedback and users' expectations of MAJAN. Participants were happy with most of the proposed concepts, but they provided some additional requirements as well. This helped to decide the priority levels of the tasks for the implementation phase.

3.2.3 Designed Concepts

In the **Design** phase which was executed after the collection of requirements, the template coordination BTs were designed, and the initial UI mockups were updated based on the requirements. Moreover, it is important to note that the existing SPARQL-BT modeling functionalities of AJAN Editor were not updated and evaluated in this work, since MAJAN's objective did not include anything with regard to the SPARQL-BT creation technique of AJAN.

Template Coordination BTs:

In this paragraph, the designed BTs for the three proposed coordination protocols (section 3.2.1) are illustrated without going into detail. Section 2.1.3 describes the execution flow of BTs.

1. To implement the **FIPA Request protocol**, four BTs for different parts of the protocol were designed. Figure 3.7 illustrates the protocol diagram and the labels of BTs for the different steps.

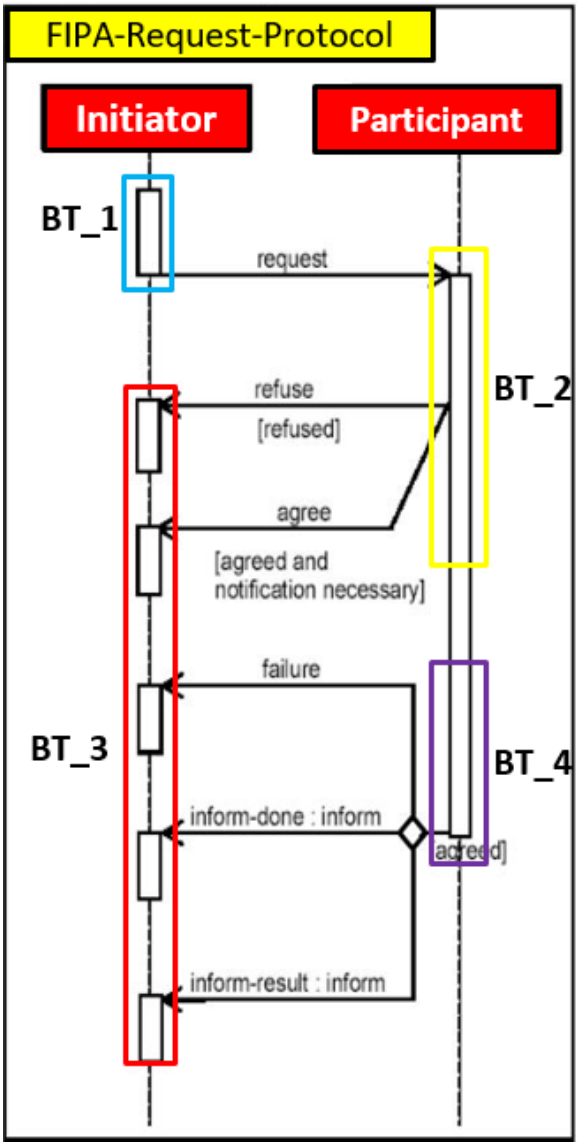

Figure 3.7: FIPA Request Protocol Diagram with BT Labels

BT_1 (in figure 3.8) was designed to send a request to participant agents as the first step of the Request protocol. The node that is labeled as *HandleInput* in this BT expects generic information as the input such as **message content**, **quorum**, **timeout**, and **notification necessary**. Once the correct input is received, the *SendMessages* node sends the request to the participants. After the message is sent, the BT waits via the *UntilSuccess* (i.e. ↻ icon) node, until the specified quorum (via *QuorumReached* node) or timeout (*TimeoutReached* node) is reached.

The *GetResponses* node loads the responses of participants so that the quorum can be checked in this BT.

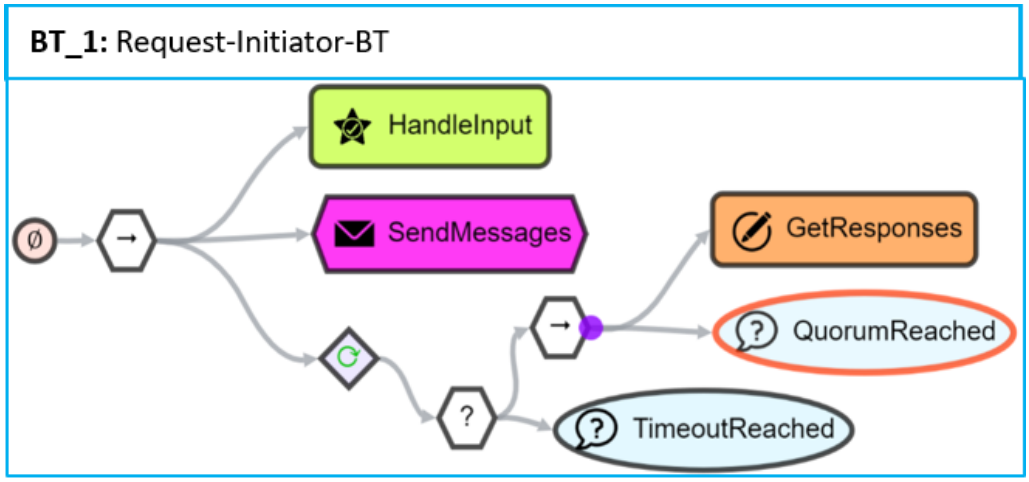

Figure 3.8: BT_1 sends a request to participants

BT_2 (in figure 3.9) was designed for the participant agents such that they can receive and handle (via *HandleInput* node) a request that should include generic information such as *timeout*, *notification necessary*, and *ID of initiator agent*. Then this BT evaluates whether it agrees or refuses the request via the condition node labeled as *Agreement*, and then it responds accordingly with either *SendAgreed* or *SendRefused* nodes. Moreover, the participant agents return a response only if the **notification necessary** field is *true* by checking it via the *NotificationNecessary* node and the **timeout** is not reached (via the *TimeoutReached* node).

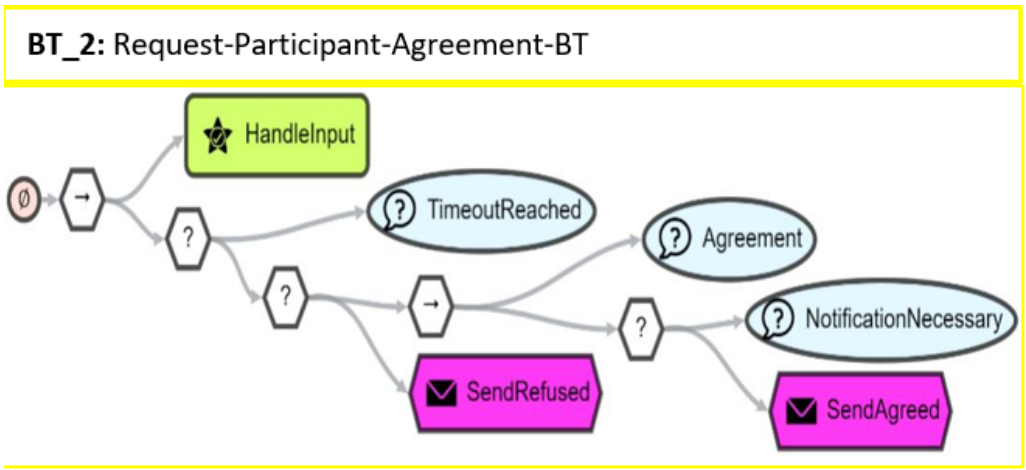

Figure 3.9: BT_2 receives a request and returns an agreement or a refusal

BT_3 (in figure 3.10) was designed for the initiator agents such that they can

receive the response of a participant in the correct format via the *HandleResponse* node and store it in the agent knowledge via the *SaveResponse* node.

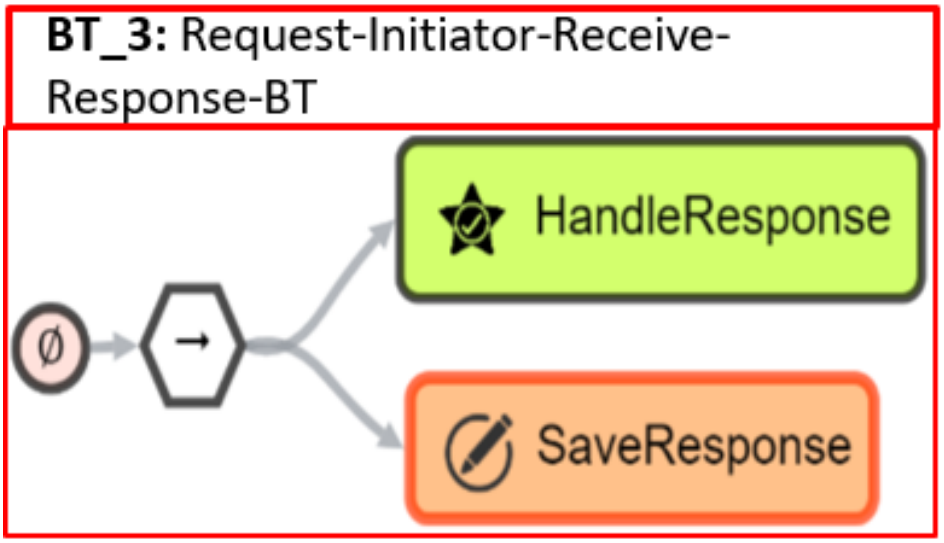

Figure 3.10: BT_3 receives and stores the responses of participants

BT_4 (in figure 3.11) was designed for the participant agents to send the result (via the *SendResult* node) for the request (that is sent by the initiator agent in BT_1) to the initiator agent only if the timeout is not reached. The *HandleInput* node checks if the BT receives the input in the correct format so that the message (in the *SendResult* node) can be formulated in a common structure that all agents can understand each other.

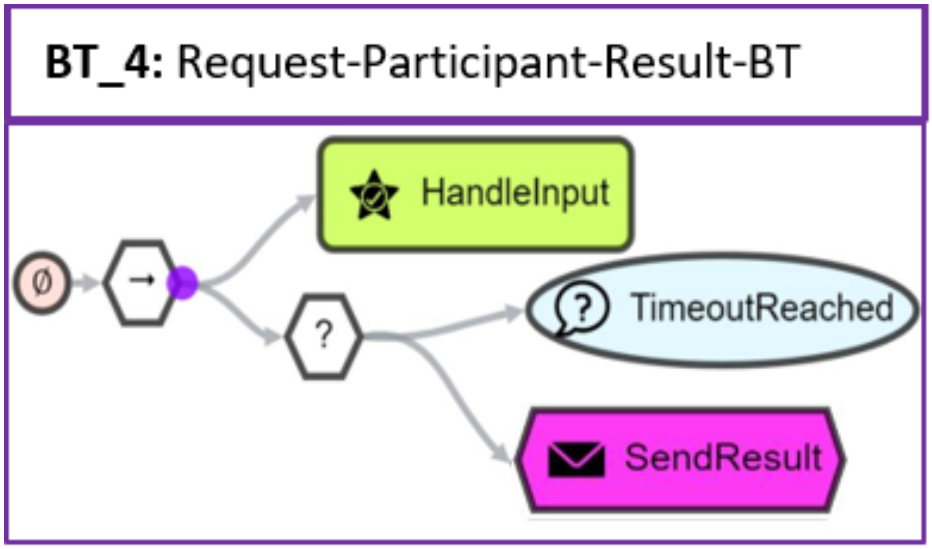

Figure 3.11: BT_4 sends the result of request to initiator agent

2. **CSGP Coordination protocol** was designed to coordinate agents into groups by solving Coalition Structure Generation Problem. In this protocol, agents share only the necessary information with each other by **using the Request protocol BTs** and then one of the agents runs a CSGP solver algorithm to find a solution (i.e. groups of agents). More specifically, the steps of this protocol (figure 3.12) are as follows:

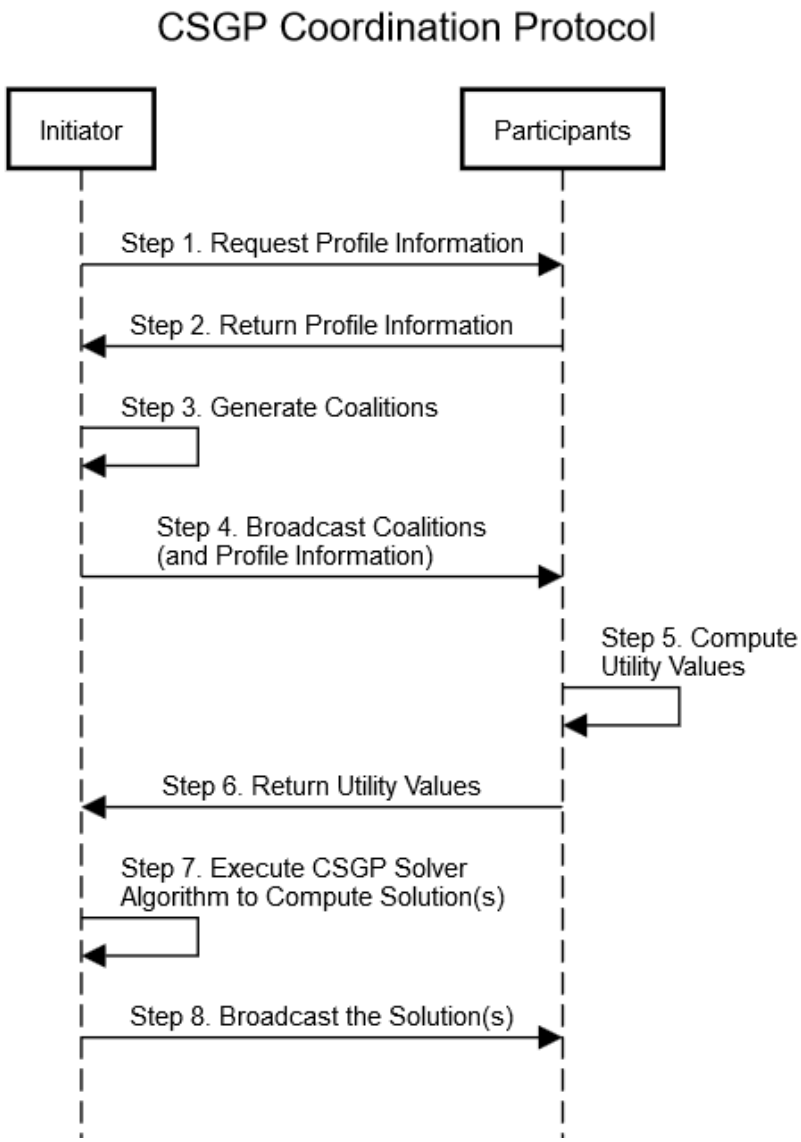

Figure 3.12: CSGP Coordination Protocol Diagram

(a) Initiator (also called dedicated) agent requests the profile information (**step 1** in figure 3.12) from the participant agents, since this information is necessary to evaluate the similarity between agents to compute groups.

(b) After the participants return their profile information (**step 2** in figure 3.12), the dedicated agent generates all possible coalitions (**step 3** in figure 3.12),

since CSGP solver algorithms require the coalitions and their values as the input.

(c) Once the coalitions are generated, the dedicated agent broadcasts them (**step 4** in figure 3.12) to the participants, together with the profile information of all agents.

(d) In **step 5** in figure 3.12, all agents compute their utility values for the coalitions they are a member of. To do so, they use their individual preferences (that are not shared) and the profile information of all agents to compute to what extent other agents satisfy their preferences.

(e) After the utility values are computed, the participants send them to the dedicated agent (**step 6** in figure 3.12).

(f) Once the dedicated agent collects the utility values, it computes the coalition values by summing up the utility values. Then it executes a CSGP solver algorithm (**step 7** in figure 3.12) and passes the coalitions and their values as the input.

(g) Finally, the dedicated agent broadcasts the solution(s) to the participants after the solver algorithm finishes (**step 8** in figure 3.12).

3. The **Clustering Coordination protocol** was also designed to create groups of agents. This protocol solves a clustering problem. Moreover, the Request protocol BTs were used to share the necessary information among agents, as in the CSGP coordination protocol. Furthermore, a clustering solver algorithm is executed by one of the agents to compute clusters of agents in this protocol. The steps of this protocol (figure 3.13) are described below:

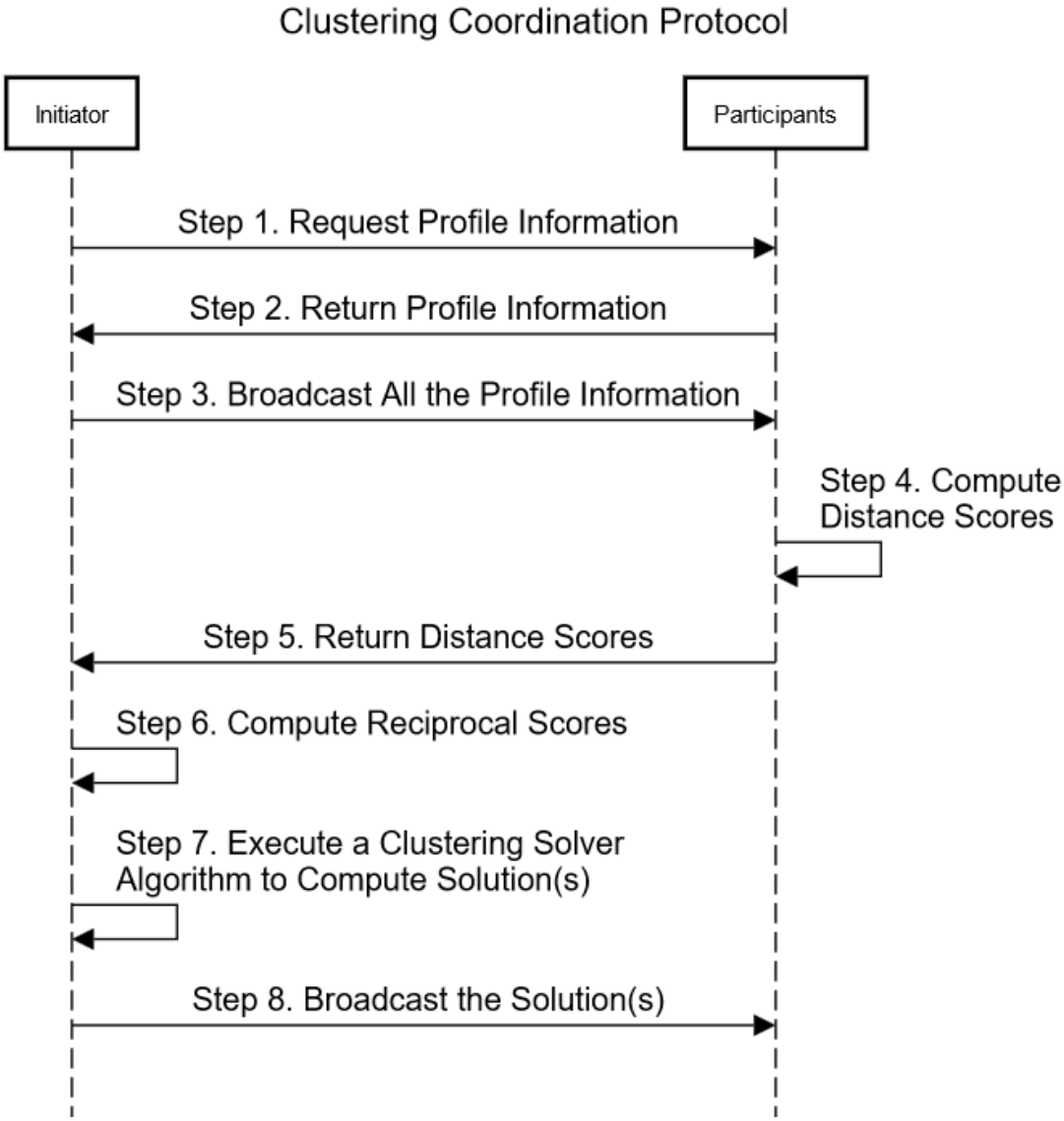

Figure 3.13: Clustering Coordination Protocol Diagram

(a) The initiator (also called the dedicated) agent requests the profile information (**step 1** in figure 3.13) from the participant agents, like in the CSGP coordination protocol.

(b) Once the profile information is collected (**step 2** in figure 3.13), the dedicated agent broadcasts it to the participants (**step 3** in figure 3.13).

(c) In **step 4** in figure 3.13, all the agents compute their distance values by using their individual preferences (that are not shared) and the profile information of other agents.

(d) After the distance values are computed, the participants send them to the

dedicated agent (**step 5** in figure 3.13).

(e) After collecting the distance values, the dedicated agent computes the reciprocal values (**step 6)** in figure 3.13) because the solver algorithms require a symmetric matrix of reciprocal values instead of an asymmetric matrix of distance values.

(f) The dedicated agent executes a clustering solver algorithm and computes a clustering solution that consists of agent clusters in **step 7** in figure 3.13.

(g) Finally, the dedicated agent broadcasts the solution to the participants (**step 8** in figure 3.13).

Monitoring Panel in MAJAN:
Nothing was updated in the Monitoring mockup since users did not specify any requirements for this functionality as stated in section 3.2.2.

Evaluation Panel in MAJAN:
Based on the collected requirements (section 3.2.2), the mockup of the Evaluation panel was updated, and it is described in figure 3.14. The modifications of each section of the Evaluation panel are elaborated in the following:

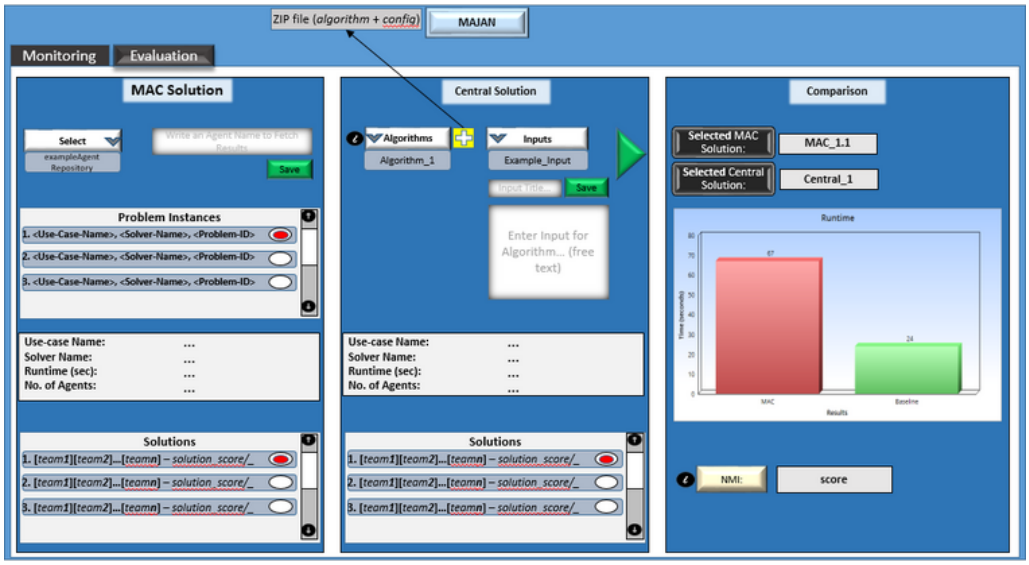

Figure 3.14: The updated version of Evaluation Panel mockup

- In the **MAC Solution** section, the hint in the edit text (next to the *Save* button) was updated to *"Write an Agent Name to Fetch Results"* to make the label self-explanatory.
- In the **Central Solution** section, the concept of uploading a central algorithm was updated, as the users required to be able to read the information about the selected algorithm. In the updated concept, it is necessary to upload a configuration file of the algorithm. This file should contain information about the *title of the algorithm, the number of agents, and the input structure* that the algorithm requires.

Then MAJAN parses the configuration file and displays the information about the selected algorithm when the *i* button next to the *Algorithms* in figure 3.14 is clicked.

- In the **Comparison** section, the UI was modified in a way that the selected solutions are displayed in the **Selected MAC Solution** and **Selected Central Solution** fields. Moreover, a bar chart was added to the UI since it was the preferred chart by users.

3.2.4 Implementation of Features

In the Implementation phase, the concepts that were created and presented previously in section 3.2.3 were implemented. This section presents the technologies that were used for the implementation, before describing the implementation details of the concepts.

General Implementation Information:

As stated in section 2.1, AJAN Service and AJAN Editor were developed using **JAVA** and **Ember.js** [81]. Since these two software products were extended in this work, the same technologies were also employed for the development of MAJAN. Ember.js is a JavaScript framework for building web applications, and it provides lots of features to build rich web UIs [81]. Moreover, JavaScript [21], HyperText Markup Language (HTML) [71], and Cascading Style Sheets (CSS) [29] were also used as they are necessary for web development. By using these technologies, the **MAJAN Plugin** and **MAJAN Web** were implemented and integrated into AJAN Service via its plugin system and to AJAN Editor, respectively.

- **MAJAN Plugin** provides the coordination nodes such as **Broadcast**, **BOSS**, **HDB-SCAN**, **CoalitionGenerator**, and **Insert**. These nodes are available in AJAN Editor for users to design SPARQL-BTs.
- **MAJAN Web** was directly integrated into AJAN Editor, and it provides the Monitoring and Evaluation concepts that were presented in the previous sections.

The guideline document in the MAJAN GitHub repository [16] gives detailed instructions for using these features.

All the functionalities that were described in the Design phase were implemented and worked as aimed. MAJAN web was connected to the back end such that the required data was fetched dynamically without hard-coding anything. Furthermore, the source code of the implementations together with the guideline document and more are provided publicly in the aforementioned MAJAN GitHub repository.

Implementation of Template Coordination BTs:

One of the most important concepts that MAJAN provides is the template coordination BTs. In this respect, thirteen BTs were developed in total to execute the FIPA Request, CSGP, and Clustering coordination protocols as stated in section 3.2.3. All these BTs can be found and customized in the **Behaviors** tab in AJAN Editor as shown in figure 3.15.

[16] `https://github.com/AkbarKazimov/MAJAN`

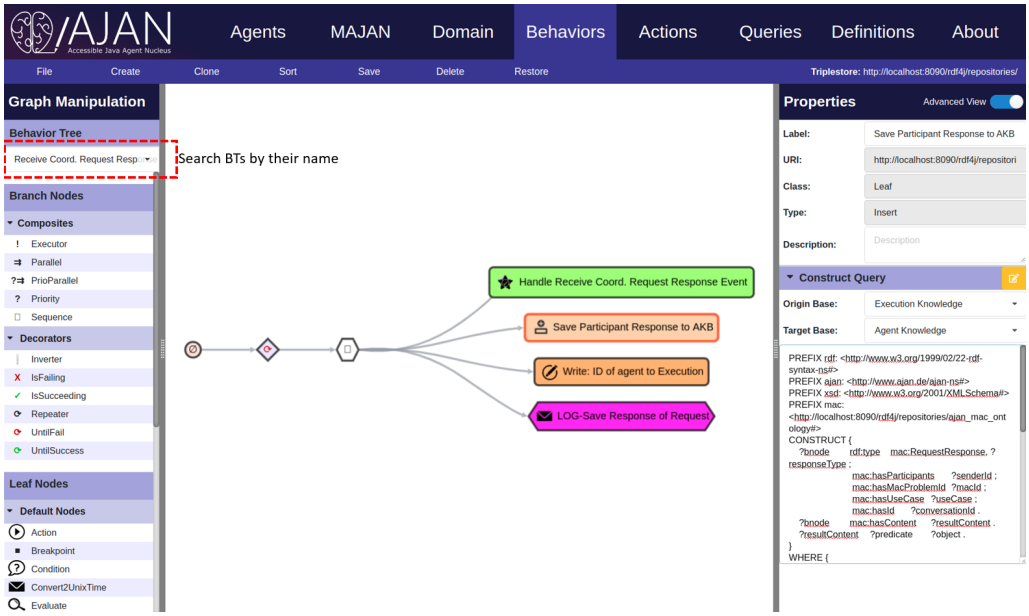

Figure 3.15: Behaviors tab in AJAN Editor to model behaviors of agents

One of those thirteen template BTs was reused in all three protocols and two of them were reused in the CSGP, and Clustering protocols. Moreover, as intended in section 3.2.3, the coordination BTs for the CSGP and clustering protocols were developed by extending and customizing the FIPA Request protocol BTs as they are generic and can be adapted to different use-cases.

Since the details of the coordination BTs are provided in the MAJAN Guideline document that can be accessed in the MAJAN GitHub repository [47], this section does not get into the details. Nevertheless, the Request protocol BTs are briefly presented below.

Moreover, all the coordination BTs ended up being more complex and bigger after the implementation phase than they were planned in the Design phase (section 3.2.3). That is because, when modeling the BTs and the coordination of agents in practice, there were lots of details that must be considered carefully to make them as resistant and generic as possible.

Three BTs were developed to implement the FIPA Request protocol. The **first BT** is called *Send Coordination Request BT* (figure 3.16), and it sends a message to request the profile information from other agents via the message node that is labeled as *"Send Request Messages to Participants"* and it waits (via the *UntilSuccess* node, ↻ icon) until either the quorum or timeout is reached which are checked with the *Is Quorum Reached?* and *Is Timeout Reached?* condition nodes. This is the BT_1 (figure 3.8) that was presented in the Design phase.

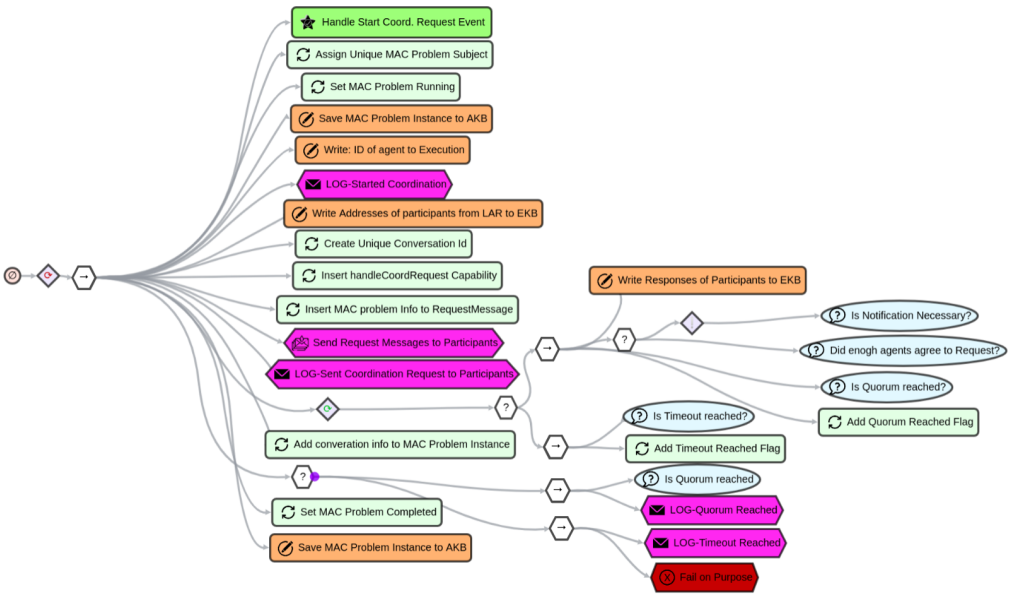

Figure 3.16: Send Coordination Request BT of Request Coordination protocol

Moreover, a unique ID for each instance of the coordination process is set via the node labeled as *Assign Unique MAC Problem Subject*, to allow the agents to run multiple coordination processes in parallel. Otherwise, the ID of different coordination processes might be the same, which would cause the processes to fail or work incorrectly.

Finally, after the MAC process is completed (i.e. node *Set MAC Problem Completed*), the BT saves all the information about this MAC problem to its Agent Knowledge Base (AKB) via the *Save MAC Problem Instance to AKB* node as shown in figure 3.16.

The **second BT** (figure 3.17) is named *Handle Coordination Request BT*, and it receives the request sent by the *Send Coordination Request BT* (in figure 3.16). The initial step in this BT is to decide whether the received request should be agreed upon or refused. To do so, the BT can be customized by users based on the needs of the use-case at hand by replacing the *Wait 3 seconds* node which is used as a placeholder.

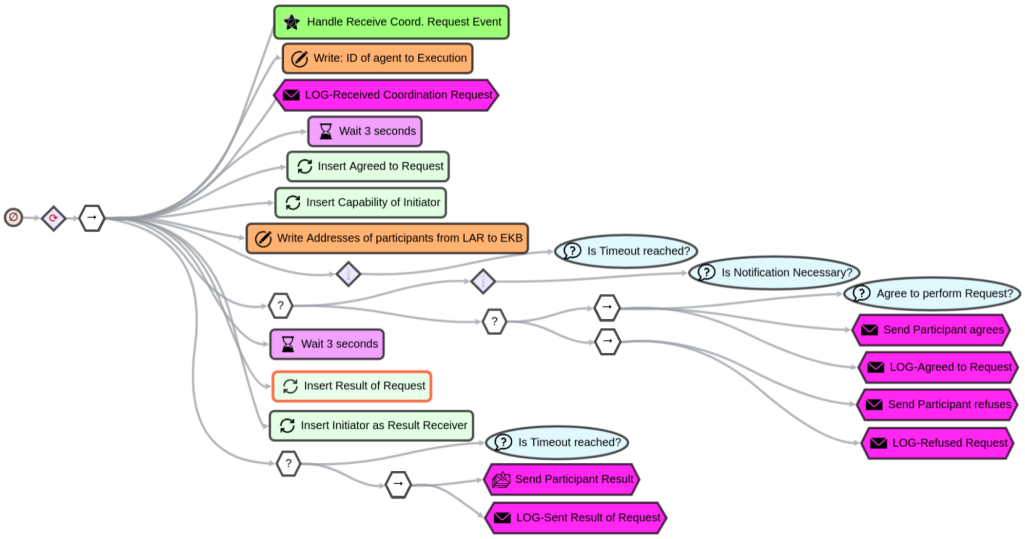

Figure 3.17: Handle Coordination Request BT of Request Coordination protocol

In this BT (figure 3.17), the *Insert Agreed to Request* node is used to agree to the request, which is checked by the *Agree to perform Request?* condition node. Consequently, the agent sends an **Agreement** via the *Send Participant Agrees* node only if it is necessary to return a notification, which is checked by *Is Notification Necessary?* node.

In the end, this BT (figure 3.17) returns the result of the request via the *Send Participant Result* message node. Moreover, BT_2 (figure 3.9) and BT_4 (figure 3.11) were combined and covered in this single BT, since it is more convenient to use and customize one BT.

The **third BT** (figure 3.18) is called *Receive Response for Coordination Request*, and it is used to receive and store the responses of other agents in the AKB such that those responses can be retrieved and used in different BTs. This BT is the implemented version of BT_3 (figure 3.10) from the Design phase.

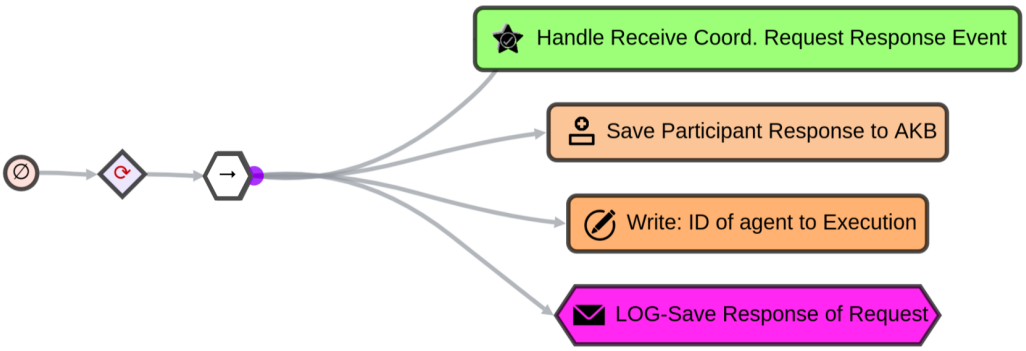

Figure 3.18: Save Coordination Response BT of Request Coordination protocol

Additionally, all the represented BTs log the coordination activities to the Monitoring panel by using LOG messages.

Implementation of Monitoring Concept:
The **MAJAN** tab was integrated into **AJAN Editor**, in between the **Agents** and **Domain** tabs as shown in figure 3.19.

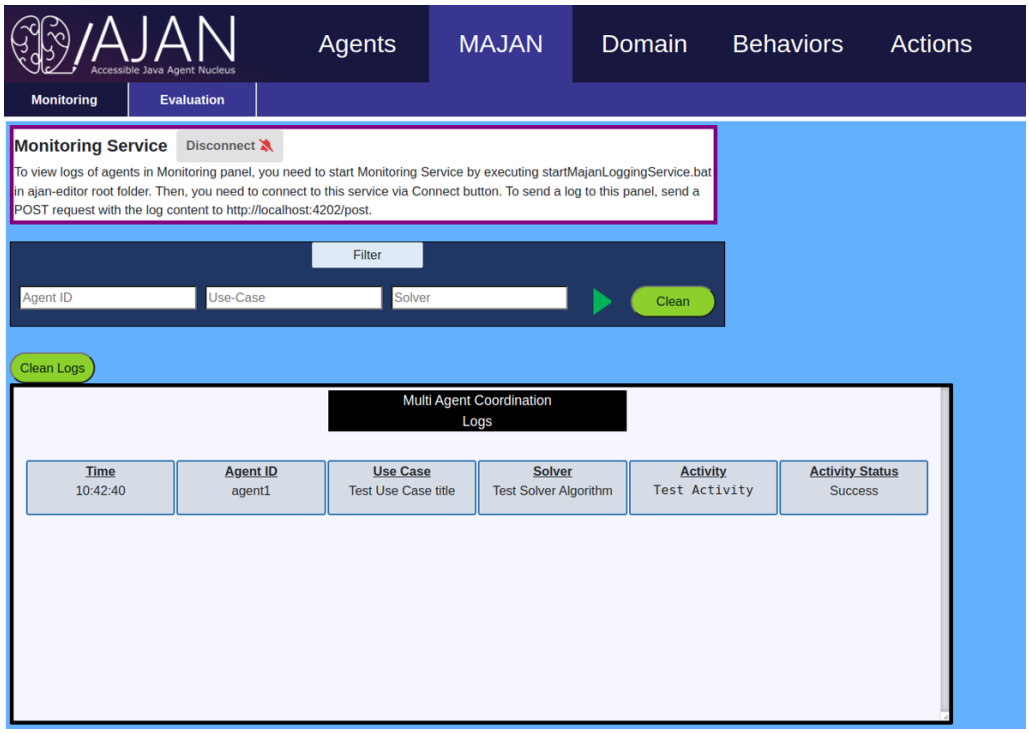

Figure 3.19: Implemented Monitoring panel in MAJAN

The idea of the Monitoring panel was to provide an endpoint to be used by the agents to send their coordination logs. However, Ember.js does not support providing an endpoint. Therefore, the web socket technology was used to implement this feature. To do so, a service that runs on a specific port was created by using JavaScript. Then the agents send their logs to this service so that the logs can be displayed in MAJAN. In the meantime, MAJAN constantly listens to this service (after *Connect* button in figure 3.19 is clicked), reads the logs from there, parses them, and displays them nicely in the UI as shown in figure 3.19.

In order for MAJAN to be able to parse the logs, a common message structure (as shown in figure 3.20) was created. This structure should be followed by all agents when sending their logs. In this message, agents should specify their *ID*, *a use-case name*, *a solver algorithm name*, *an activity* that they performed, and *the status of this activity* as described in figure 3.20 in lines 4, 5, 6, 7, and 8, respectively.

```
1. @prefix mac: <http://localhost:8090/rdf4j/repositories/ajan_mac_ontology#
> .
2. @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#
> .
3. _:bnode     rdf:type     mac:Log ;
4.             mac:hasAgentId      "<agentID>" ;
5.             mac:hasUseCase      "<useCaseTitle>" ;
6.             mac:hasSolver     "<solverName>" ;
7.             mac:hasActivity "<activityContent>" ;
8.             mac:hasStatus     "<activityStatus>" .
```

Figure 3.20: Common Structure of MAJAN Logs

Additionally, the filter function allows users to filter the logs based on agent ID, use-case title and/or solver name.

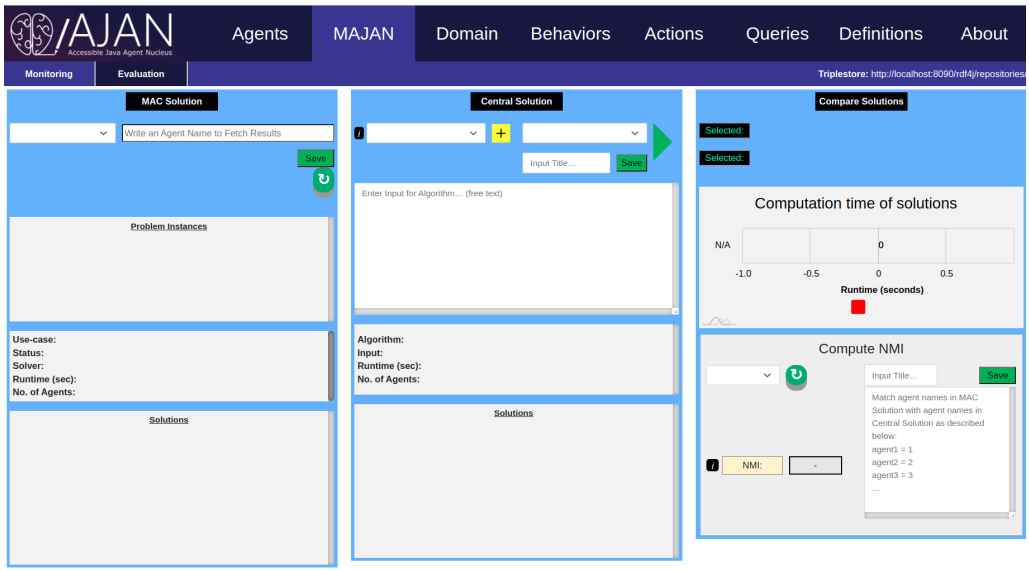

Figure 3.21: The implemented version of the Evaluation panel in MAJAN

Implementation of the Evaluation concept:

This paragraph explains the implementation details of the following three functionalities that are provided in the Evaluation panel (in figure 3.21):

1. Display the grouping solutions of a MAC problem in the **MAC Solution** section.
2. Compute the grouping solutions by executing a centrally running grouping algorithm and display them in the **Central Solution** section.
3. Compare the solutions of MAC and central algorithm to validate whether the multiagent system and central executions produce the same solutions in the **Compare Solutions** section.

In the **MAC Solution** section, an agent name should be written and saved by the users in the *"Write an Agent Name to Fetch Results"* field so that MAJAN can retrieve the results from the specified agent and display them as shown in figure 3.22.

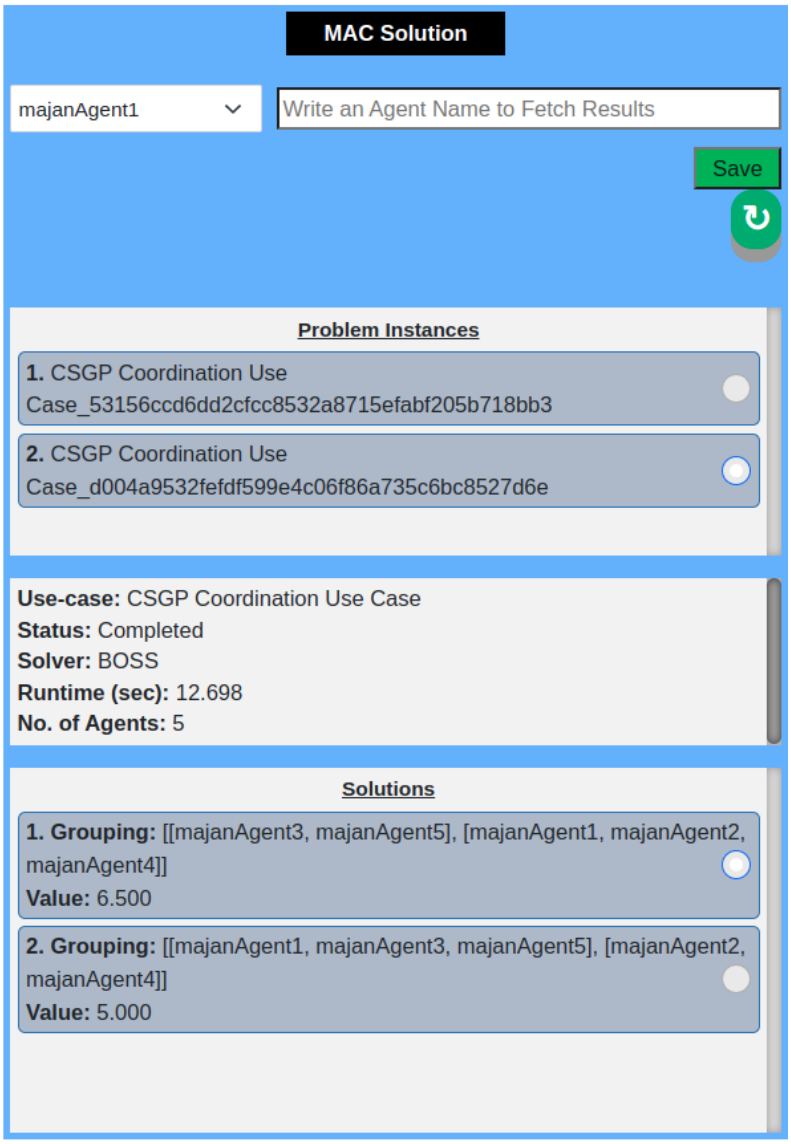

Figure 3.22: MAC Solution section in MAJAN

Moreover, it is required to store the MAC results in the agent knowledge in a certain format such that MAJAN can parse it. To do so, an ontology was created to describe a MAC problem and its solutions, as illustrated in figure 3.23. According to the ontology, every MAC problem should have an ID (line 6), use-case title (line 7), status (line 8), solver name (line 9), runtime in seconds (line 10), number of agents (line 11), and a grouping solution (line 12). Additionally, each grouping solution should have groups

(line 15). Moreover, each solution might have a value (line 13) and a rank (line 14). Finally, a group should have member agents (line 16).

```
1. PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2. PREFIX ajan: <http://www.ajan.de/ajan-ns#>
3. PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
4. PREFIX mac: <http://localhost:8090/rdf4j/repositories/ajan_mac_ontology#>
5. _:macProblemBnode    rdf:type          mac:MACProblemInstance ;
6.                      mac:hasId         "<uniqueID>" ;
7.                      mac:hasUseCase    "<useCaseTitle>" ;
8.                      mac:hasStatus     "<macStatus>" ;
9.                      mac:hasSolver     "<solverName>" ;
10.                     mac:hasRuntime <runtimeInSeconds> ;
11.                     mac:hasNumberOfAgents    <numberOfAgents> .
12. _:macProblemBnode   mac:hasSolution _:groupingBnode .
13. _:groupingBnode     mac:hasValue      <groupingValue> ;
14.                         mac:hasRank <groupingRank> ;
15.                                         mac:hasMembers   _:groupBnode .
16. _:groupBnode        mac:hasMembers    <groupMemberAgent> .
```

Figure 3.23: The format to store a MAC problem and its solutions in AKB

The **Central Solution** section (figure 3.24) allows users to upload a JSON configuration file of a central algorithm using the yellow plus button, to provide input to this algorithm, execute it, and display its results. The configuration file should include the following information:

- Name of the algorithm
- Path to the JAR file of the algorithm
- Timeout value that describes the amount of time for MAJAN to wait until stopping the execution of the algorithm, since the algorithm might get stuck in a loop.
- Description of input structure that the algorithm expects.

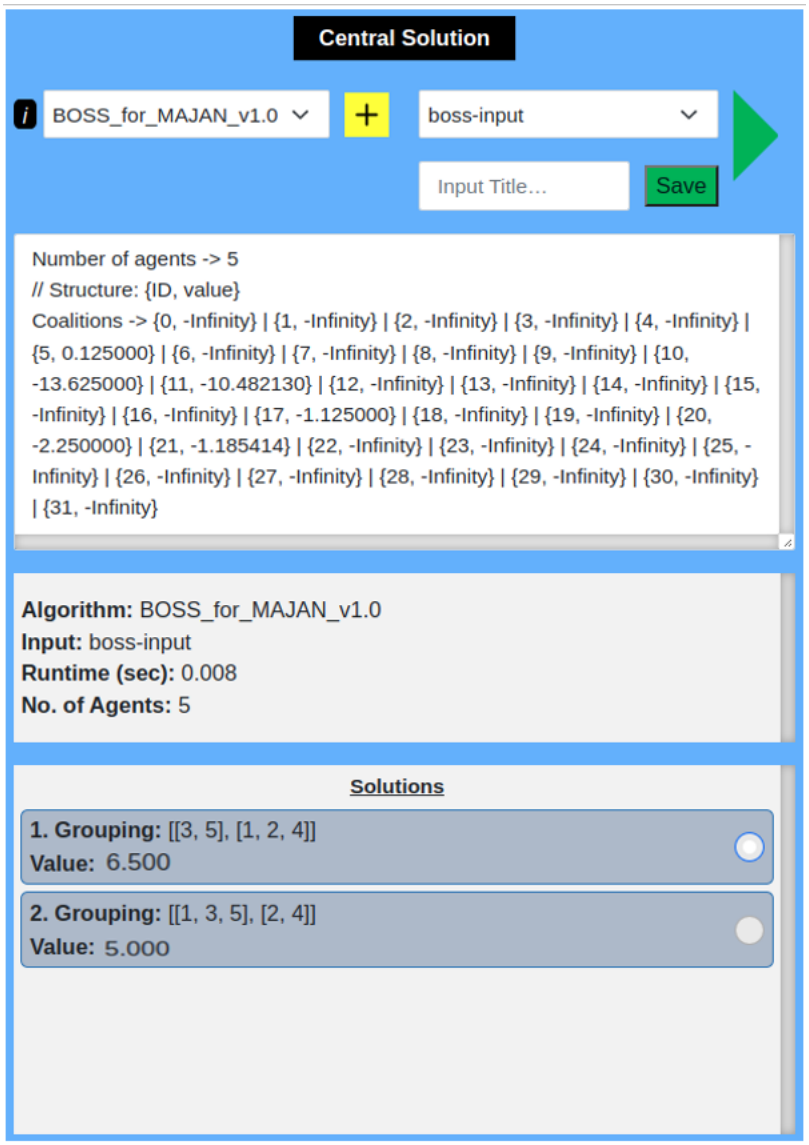

Figure 3.24: Central Solution section in MAJAN

Since each algorithm requires input to compute a solution, MAJAN enables users to add the input in the *Enter input for Algorithm... (free text)* field and save it for further use. After selecting an algorithm, an input, and clicking on the **Run** button, MAJAN executes the selected algorithm and displays its result in the **Solutions** view (figure 3.24).

In order to execute a JAR file of the algorithm, AJAN Service was extended to provide an endpoint, since Ember.js does not support running Command Prompt (CMD) commands, and consequently, it cannot run a JAR file. This endpoint expects 1) the path to the JAR file and 2) a timeout value to run the algorithm, and returns the result of the algorithm to MAJAN. Although the features of this section were designed slightly different in the Design phase (figure 3.6), during the implementation I realized that there

is no need to upload a JAR file instead its path is enough for JAVA to run it.

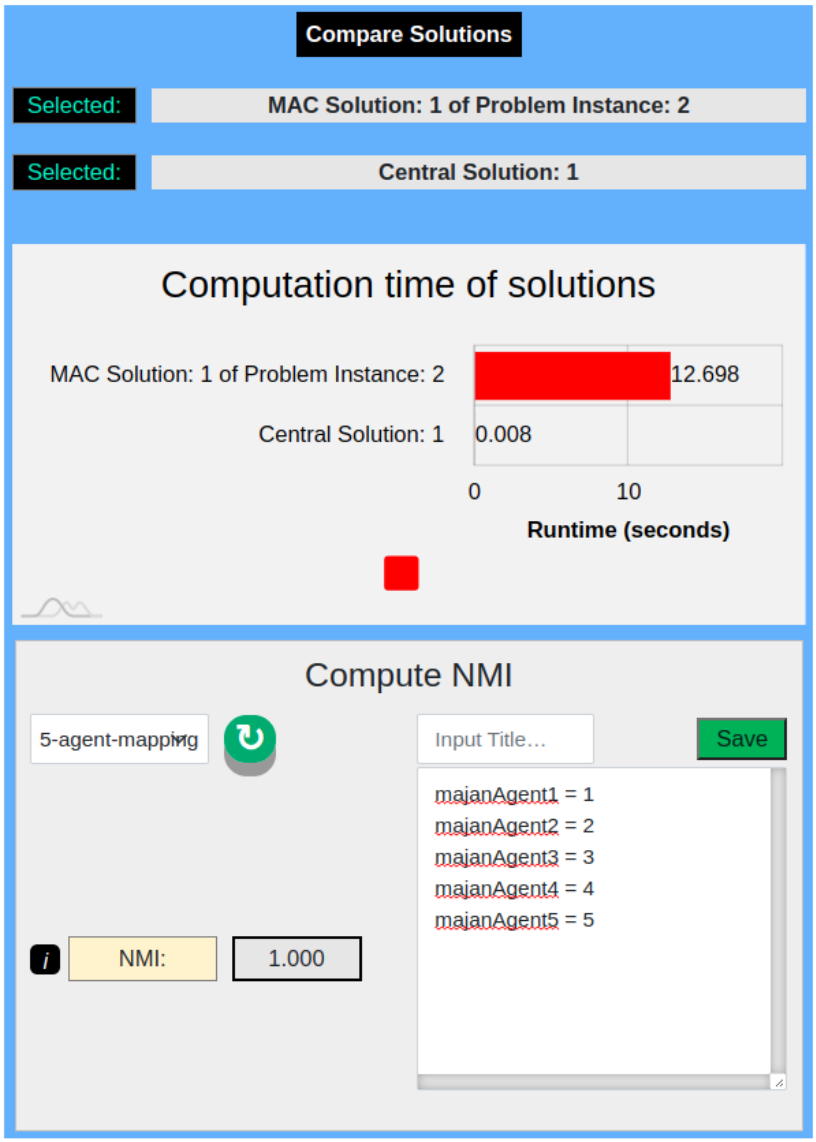

Figure 3.25: The Compare Solutions section in MAJAN

Figure 3.25 describes the UI of **Compare Solutions** section. This section allows users to view the names and the computation time of the selected solutions. In order to display a bar chart, the *amcharts5* [44] library was used since others such as *ember-cli-chart* [3], *ember-highcharts* [4], *ember-anychart* [11], *ember-froalacharts* [38], and *ember-d3* [43] did not work properly. In this GitHub repository [17], there are lots of useful examples of the amchart library.

[17] `https://github.com/amcharts/amcharts4/tree/master/dist/ember/vendor/script/amcharts4/examples/javascript`

The Compare Solutions section allows users to compute the **Normalized Mutual Information (NMI)** [35] score between the solutions. To do so, users need to provide a mapping file that maps the names of agents in the selected MAC solution to the names of agents in the selected central solution. With this mapping, MAJAN is able to match the groups correctly. Otherwise, the grouping results could not be compared since they could contain different member agents. The mapping is the list of MAC agent names on the left and central agent names on the right, as shown in figure 3.26.

```
agent1 = 1
agent2 = 2
agent3 = 3
...
```

Figure 3.26: Mapping file structure to compute NMI score

Additional Implementations:

Moreover, several Postman [1] collections were prepared to help the users to use the system easily. These collections allow creating multiple agents with one click, starting coordination protocols by running the corresponding postman request and populating the repositories of agents easily. More detailed information about them is provided in the guideline document.

3.2.5 System Usability Evaluation

The results of the Evaluation phase are provided in this section. As discussed in section 3.1.5, the evaluation survey started with the background questions (section B.1), followed by tasks (section B.2) to be performed by participants, continued with System Usability Scale (SUS) questions (section B.3) and finalized with the complementary questions (section B.4) to collect the final feedback. The questions and the tasks of the survey are provided in appendix B.

Two expert and three non-expert participants who were unbiased and familiar with AJAN participated and completed the evaluation survey with success. The SUS score of the system is provided at the end of this section.

Feedback from the Participants

The following feedback was collected from the participants' answers and by observing their actions while they perform the tasks:

1. MAJAN README (which is required for MAJAN GitHub repository [18]) and MAJAN Guideline documents were confusing.
2. The meaning of the *quorum* was not completely clear in the guideline document.
3. The *Coalition*, *Coalition Structure*, and *CSGP* were not completely understandable in the guideline document.
4. The results were not fetched automatically after an agent name was saved in the MAC Solution section.
5. The *Selected* solutions view in Compare Solutions section was not self-explanatory.
6. The status of the MAC process was hard to find in the logs in the Monitoring panel.

[18] `https://github.com/AkbarKazimov/MAJAN`

7. The *Connect* button was not understood whether it was connected or not.
8. The last log was displayed at the end of the list in the Monitoring panel, but it would be better if the last log was at the top of the list.
9. *Using MAJAN* title was misleading in the guideline document.
10. The color scheme and buttons in the UI of MAJAN could be improved such that it is the same as in AJAN Editor.
11. A nice-to-have feature was to sort the logs based on the time in the Monitoring panel.

System Usability Scale Score:
In the survey, the participants rated the system by answering the standardized System Usability Scale (SUS) [22, 45] questions on a scale from 1 (strongly disagree) to 5 (strongly agree). The SUS questions and their average rates are provided in section B.3.

SUS score	Adjective
≥ 84.1	Best Imaginable
≥ 80.8	Excellent
≥ 71.1	Good
≥ 51.7	OK
≥ 25.1	Poor
≥ 0	Worst Imaginable

Table 3.1: SUS score interpretation according to [45]

Based on the rates, the SUS score of MAJAN was computed as 83.5 which corresponds to *Excellent* in the SUS table 3.1. This means that the system has high usability and understandability, but there is room for further improvements.

3.2.6 Final Version of MAJAN

Although a second iteration of the selected life cycle model was not executed because of time constraints, the implemented MAJAN features were updated based on the feedback that was collected in the Evaluation phase. In this respect, the first 9 of 11 requirements listed in 3.2.5 were implemented. The final version of the MAJAN Monitoring and Evaluation panels are shown in figures 3.27 and 3.28, respectively.

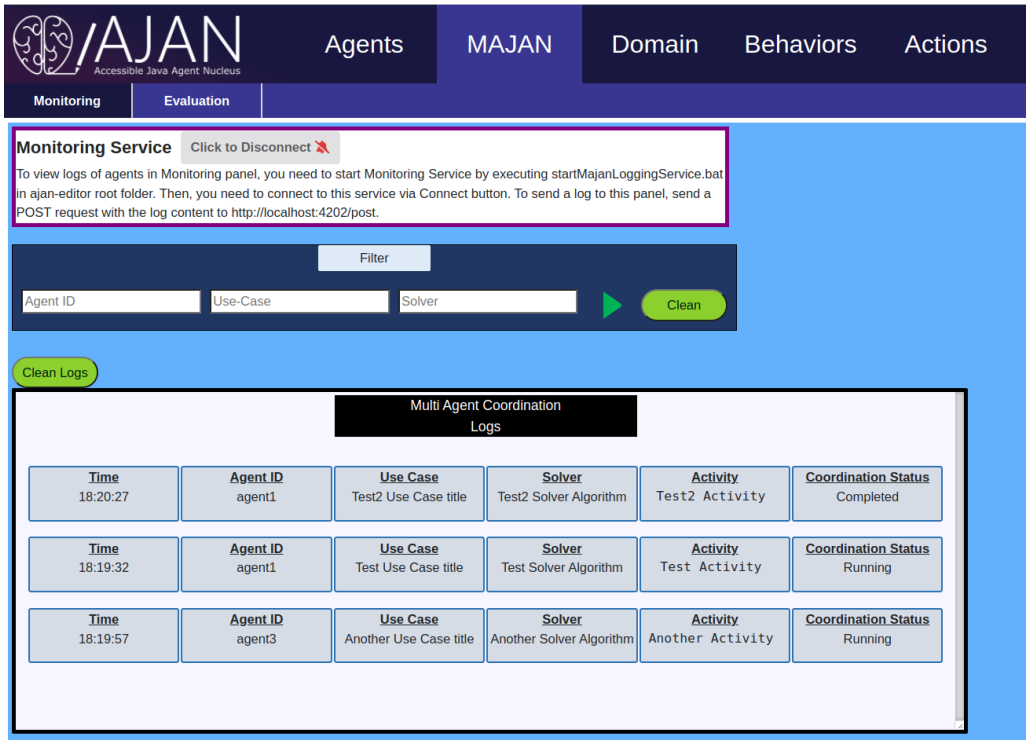

Figure 3.27: Final version of Monitoring panel in MAJAN

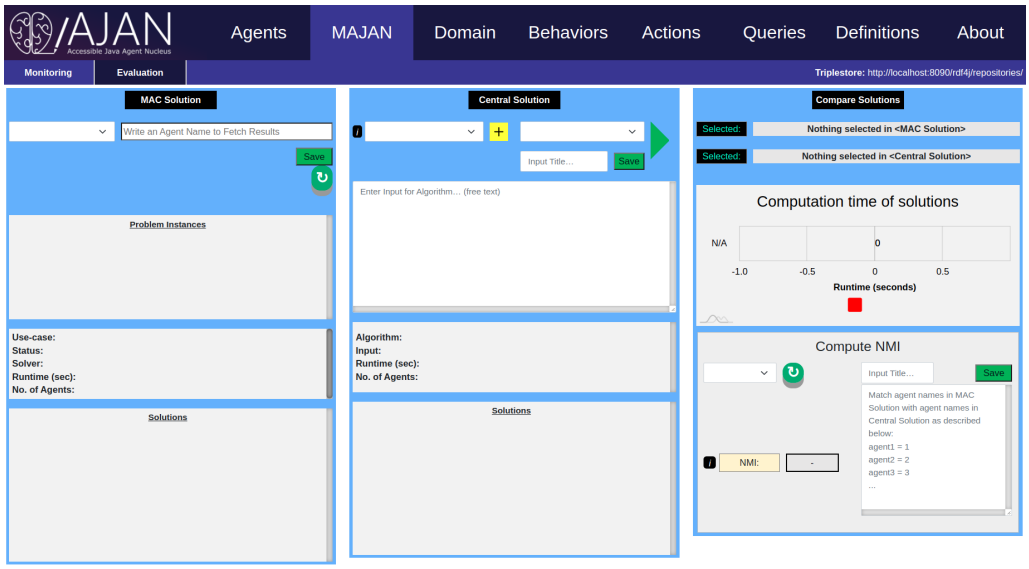

Figure 3.28: Final version of Evaluation panel in MAJAN

Chapter 4
Language Course Coordination Use-Case

This chapter presents the given MyWELCOME use-case Language Course Coordination (LCC) (section 2.2.3) and its implementation with MAJAN. More specifically, its description, analysis of the grouping algorithms to solve it, its implementation, and evaluation results are provided in the following sections.

4.1 LCC Description

The LCC use-case was concerned with helping the language course instructor to divide the students in a classroom into smaller working groups to improve the efficiency of language learning. To do so, MyWELCOME agents needed to coordinate among each other to form an (approximately) optimal groping of TCNs for each lesson of the course based on the given individual (preferences) and the system constraints of the use-case and the individual Course Progress Level (CPL) of TCNs. Finally, the result(s) should be recommended to the course instructor so that he/she could select a final grouping among them in which the TCNs were most suitable for each other. The suitability of the result could be determined by the degree of satisfaction of the individual and system constraints. Therefore, the objective of LCC for the MyWELCOME agents was to maximize the suitability of groups by satisfying as many as constraints while also taking into account the CPL of TCNs.

All the constraints of LCC were listed in section 2.2.3. Below, the additional requirements that were provided by the users during the development phase are described:

- Functionality for the TCNs to specify the preferences that they prioritize more. To realize this, TCNs were allowed to assign an importance weight to each individual preference through the MyWELCOME application. The value of the importance weight ranged between 0 (not important) and 1 (most important). According to PRAKSIS, gender preference should be considered three times more important than nationality preference if the TCN did not specify any weights.

- Functionality for the course instructor to specify the importance of satisfaction of individual preferences compared to the similarity of CPL of TCNs in a group. For this purpose, the course instructor was allowed to assign an importance weight through Teacher Panel (TP) before he/she started the coordination to groups, such that agents could use this importance weight in their computation of the groups of TCNs. The value of this importance weight also ranged between 0 (not important) and 1 (most important).

4.2 LCC Conceptual Solution

This section discusses the solution of LCC. Firstly, the grouping algorithms from section 2.3.1 are analyzed to determine the ones that can solve it. Then, the coordination protocol for LCC is explained. After providing the implementation details, the evaluation results are presented.

4.2.1 Analysis of the Grouping Algorithms for LCC

To determine the possible solver algorithms for LCC, the first step was to identify the algorithmic requirements as follows:

1. The solution of LCC should be a **set of groups (i.e. a grouping)** based on its description in section 2.2.3.
2. The **constraints of LCC** should be considered during the computation of the solution.
3. All the TCNs in LCC should be assigned to groups. In other words, there should be **no outliers** in the solution.
4. Because of the group size constraint (item 2b in section 2.2.3) of LCC, the algorithm should be able to accept **a minimum and a maximum group size** as the input.
5. The solution of LCC should be optimal or deterministic.

Algorithm	Outputs set of groups	Supports constraints	Produces outliers	Accepts Min&Max group size	Optimal/ Determenistic	Other
K-Means (McQueen, 1967)	Yes	No	No	Yes	Yes	-
COP-KMeans (Wagstaff et al, 2001)	Yes	Yes	No	Yes	Yes	No solution if a constraint is violated
Hierarchical Clustering (Nielsen, 2016)	Yes	No	Yes	Yes	Yes	-
DBSCAN (Ester et al, 1996)	Yes	No	Yes	No	Yes	-
ACE (Alqurashi et al, 2019)	Yes	No	No	No	Yes	No available implementation
Recommender Systems (Potts, et al., 2018), (Palomares et al, 2021)	No	-	-	-	-	-
BCOP-KMeans (Okabe et al, 2018)	Yes	Yes	No	Yes	Yes	Implementation in Matlab
CoExDBSCAN (Ertl et al, 2020)	Yes	Yes	Yes	No	Yes	No solution if a constraint is violated
ACS (Taguelmimt et al, 2021)	Yes	Yes	No	Yes	No	-
HDBSCAN* (Campello R. J. et al, 2015)	Yes	Yes	Yes	No	Yes	-
BOSS (Aknine et al, 2021)	Yes	Yes	No	Yes	Yes	Implementation in JAVA

Figure 4.1: Comparison of Grouping algorithms for LCC

Based on the requirements above, the grouping algorithms (section 2.3.1) are analyzed and summarized in figure 4.1:

1. **K-Means** algorithm (item 1a in section 2.3.1) ticked all the boxes except supporting constraints (requirement 2). Therefore, it was ruled out.
2. **COP-KMeans** algorithm (item 1b in section 2.3.1) was able to solve LCC since it met all the requirements above, but it is not able to compute any solution if any of the given constraints are violated. Therefore, it was also ruled out.
3. **Hierarchical Clustering** (item 1g in section 2.3.1) was ruled out since it does not

support constraints (requirement 2), and it produces outliers (requirement 3).

4. **DBSCAN** algorithm (item 1d in section 2.3.1) does not support constraints (requirement 2), produces outliers (requirement 3), and is not able to accept maximum cluster size (requirement 4). Thus, it was not a fit for LCC.
5. **ACE** algorithm (item 1h in section 2.3.1) was also not able to solve LCC since it does not support constraints (requirement 2), and it is not able to accept the minimum and maximum group sizes (requirement 4).
6. **Recommender Systems** (item 3 in section 2.3.1) were not a fit to solve LCC since they do not produce a grouping result (requirement 1).
7. **BCOP-KMeans** algorithm (item 1c in section 2.3.1) met all the algorithmic requirements and thus, it could solve the LCC use-case. However, this algorithm had to be implemented in JAVA from scratch since AJAN (section 2.1) and MAJAN (section 3.2.4) were developed using JAVA, but BCOP-KMeans implementation was in Matlab.
8. **CoExDBSCAN** algorithm (item 1e in section 2.3.1) supports constraints unlike DBSCAN but it also produces outliers (requirement 3), and it does not accept maximum group size (requirement 4) as input. Moreover, it computes no solution in case any of the constraints is not satisfied. Therefore, it was not able to solve LCC.
9. **ACS** algorithm (item 2b in section 2.3.1) could solve LCC but its solution is not guaranteed to be optimal (requirement 5). Thus, this option was also ruled out.
10. Since **HDBSCAN*** algorithm (item 1f in section 2.3.1) produces outliers (requirement 3) and does not accept maximum group size (requirement 4), it could not solve LCC.
11. **BOSS** algorithm (item 2a in section 2.3.1) met all the algorithmic requirements and its implementation is available in JAVA. Therefore, it was the best option to solve the LCC use-case.

According to the analysis above, BOSS was selected to solve LCC. Alternatively, BCOP-KMeans and ACS could solve it if the former was implemented in JAVA and the approximate solution of the latter was acceptable for LCC users.

Projection of LCC to CSGP:

When reducing LCC to a CSG problem, a working group in LCC corresponded to a coalition in CSGP (item 2 in section 2.3.1). The CSGP maximizes the performance of a coalition and the LCC problem maximized the suitability of a group. Therefore, the coalition value was interpreted as the degree of suitability rather than the performance of the coalition. The mathematical formulations and the descriptions of the steps of solving LCC as a CSGP are described below:

1. Firstly, it was necessary to generate coalitions to solve a CSGP since the CSGP solver algorithms expect the coalitions and their values as the input as stated in section 2. When generating the coalitions, two LCC constraints were applied:
   i) the coalition/group size should be in between the range specified by the course instructor (item 2b in section 2.2.3),

$$\forall i \in \{x \in N, 1 \leqslant x \leqslant m\} : |C^i| \subseteq [S_{min}, S_{max}] \tag{4.1}$$

where m is the number of coalitions in the coalition structure, $|C^i|$ is the number of agents in coalition i, S_{min} is the minimum, and S_{max} is the maximum specified coalition sizes

ii) and there should not be more than the specified amount of students in a group who missed the previous lesson (item 2d in section 2.2.3).

$$\forall i \in \{x \in N, 1 \leqslant x \leqslant m\} : \sum_{j=1}^{|C^i|} m(C_j^i) \leqslant L \tag{4.2}$$

$$m(C_j^i) = \begin{cases} 1, & \text{if } j\text{th member of coalition } i \text{ missed the previous lesson} \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

where m is the number of coalitions in the coalition structure, C_j^i is the jth element of coalition i, L is the limit of the students who missed the previous lesson to be in the same group, and $m(C_j^i)$ is the function that returns whether the student (i.e. coalition member) missed the previous lesson or not.

2. To compute the coalition values to be passed to the solver algorithm, it was necessary to calculate utility values. Therefore, all agents computed the utility values for the coalitions that they were members of. The utility value in LCC could be defined as the degree of satisfaction of individual preferences of an agent or a TCN since each TCN was represented by an agent as stated in section 2.2.2 and they were computed by comparing the individual preferences of TCNs to the profile information of other TCNs. Moreover, a utility value started from zero, and it was increased or decreased by the amount of importance weight of this individual preference if it was satisfied or violated, respectively. The importance weight ranged between zero (not important) and one (most important). The mathematical formulation of the computation of utility values is described below.

$$\forall i \in \{x \in N, 1 \leqslant x \leqslant m\} \tag{4.4}$$

$$\forall j \in \{x \in N, 1 \leqslant x \leqslant |C^i|\} \tag{4.5}$$

$P^{C_j^i}$ is the set of individual preferences of the jth member of coalition i.
$W^{C_j^i}$ is the set of weights of the individual preferences of the jth member of coalition i.

$$p(P_k, C_j^i) = \begin{cases} 1, & \text{if the } k\text{th preference of coalition member } C_j^i \text{ is } \textbf{satisfied} \\ -1, & \text{otherwise} \end{cases} \tag{4.6}$$

$$uv(C^i, C_j^i) = \sum_{k=1}^{|P|} p(P_k, C_j^i) * W_k^{C_j^i} \tag{4.7}$$

where $\sum_{k=1}^{|W|} W_k = 1$, C_j^i is the jth member of coalition i, $p(P_k, C_j^i)$ function returns whether the given preference of a coalition member is satisfied or not, and

$uv(C^i, C^i_j)$ is the function that calculates **utility values** for the given coalition and coalition member.

3. This step calculated the coalition values, as the utility values were already computed. Since the coalition value was computed as the sum of the utility values of the members of the coalition in CSGP, the first step was to sum all the computed utility values. Additionally, the Standard Deviation (SD) [12] of CPLs for each coalition was computed to apply the LCC constraint that stated that the working groups should contain TCNs with similar CPL (item 2c in section 2.2.3). The SD is the most popular method to measure the spread of the data. Finally, the sum of the utilities and the SD value were multiplied with their respective importance weights (as explained in section 2.2.3), and then the final SD value was deducted from the final sum of the utility values to find the coalition value. The respective mathematical formulations of these computations are shown below.

$$W_{prefs} + W_{sd} = 1 \tag{4.8}$$

where W_{prefs} and W_{sd} represent the importance weights of individual preferences and Standard Deviation (SD), respectively.

$$sd(C^i) = \sqrt{\frac{\sum_{j=1}^{|C^i|} (cpl(C^i_j) - \frac{\sum_{h=1}^{|C^i|} cpl(C^i_h)}{|C^i|})^2}{|C^i|}} \tag{4.9}$$

where $cpl(C^i_j)$ gives the Course Progress Level (CPL) of the specified coalition member and the $sd(C^i)$ function calculates the SD value of the given coalition.

$$cv(C^i) = \sum_{j=1}^{|C^i|} (uv(C^i, C^i_j) * W_{prefs} - sd(C^i) * W_{sd}) \tag{4.10}$$

where the $cv(C^i)$ function calculates the coalition value.

4. The final step was to compute the coalition structure values, which were calculated as the sum of the member coalition values as described below.

$$csv(CS) = \sum_{m=1}^{|CS|} cv(CS_m) \tag{4.11}$$

where CS_m is the mth member of coalition structure CS and $csv(CS)$ is the function that computes the coalition structure values (social welfare).

The objective function of LCC (equation 4.12) is finding the optimal coalition structure $CS^* \in A$ under the constraints 4.1 and 4.2, where A denotes the set of all coalition structures [26].

$$\begin{aligned} CS^* = \arg\max_{CS \in A} csv(CS) \\ \text{subject to } \textit{constraint 4.1}, \\ \textit{constraint 4.2} \end{aligned} \tag{4.12}$$

The execution of the steps described above applied all the individual and the system constraints of LCC that are listed in section 2.2.3.

4.2.2 LCC Coordination Protocol

To solve LCC through the coordination of agents in a multiagent system, its coordination protocol was designed, and it is presented in this section with two figures. Figure 4.2 describes the interaction between the components of the WELCOME system, and figure 4.3 illustrates the interactions between the agents in their coordination process.

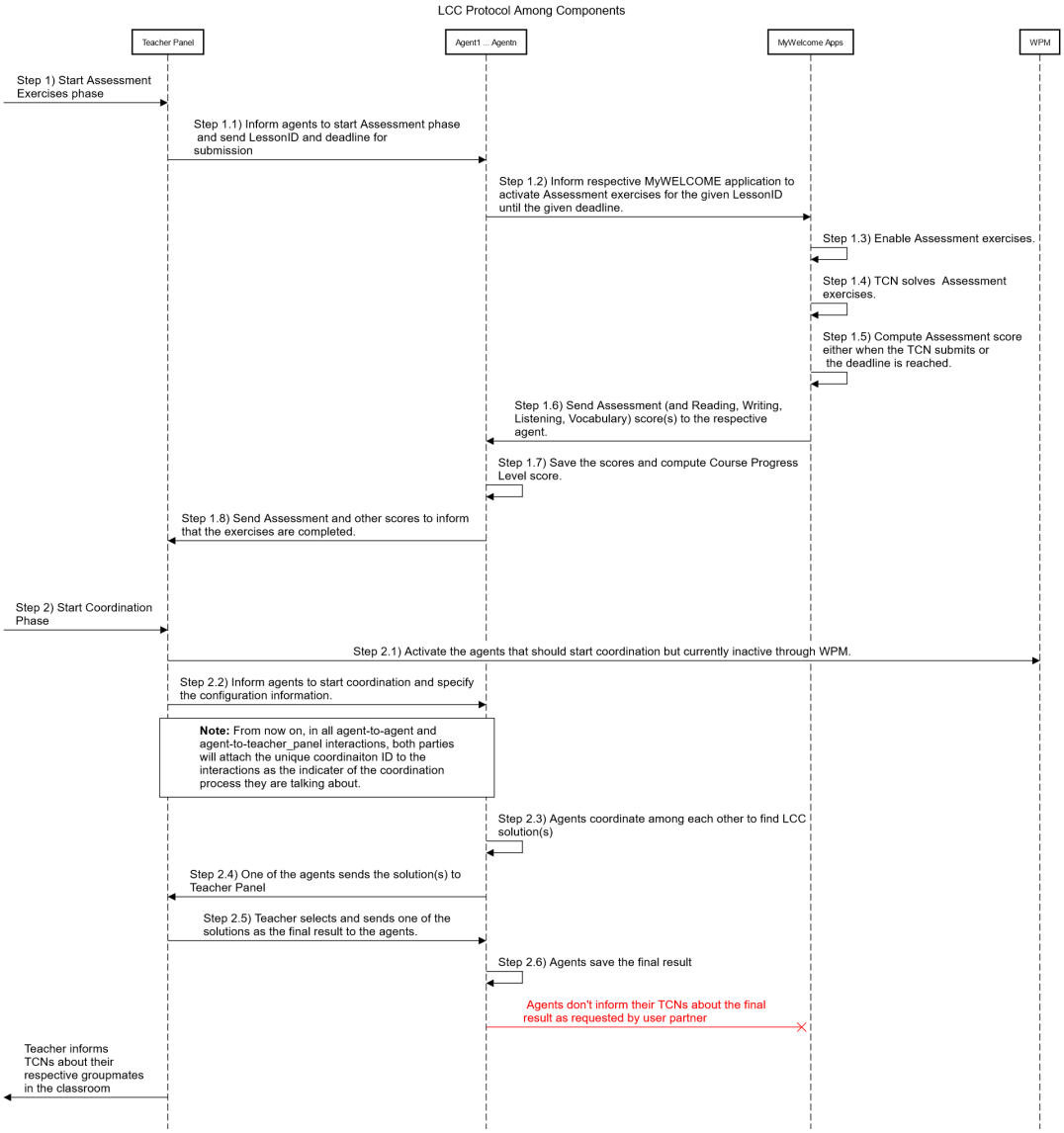

Figure 4.2: LCC interaction diagram between the WELCOME system components

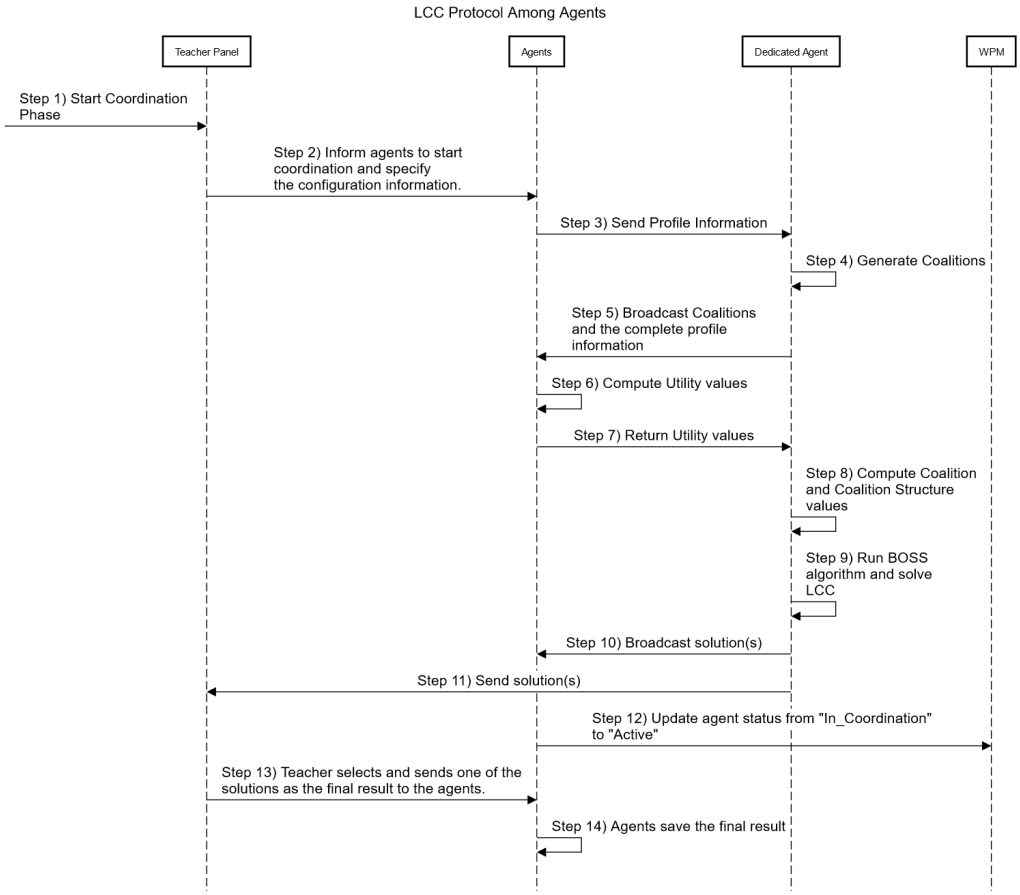

Figure 4.3: LCC coordination protocol among agents

Moreover, there were two phases in the LCC use-case as shown in figure 4.2:

1. In the **Assessment phase**, TCNs needed to perform the assessment exercises so that their assessment score can be computed and used later in the computation of the working groups.
2. In the **Coordination phase**, agents coordinated themselves into working groups to solve the use-case.

The steps of figure 4.2 are explained below:

1. The Teacher Panel (TP) (section 2.2.2) sent a message to agents (step 1.1) so that they could inform (step 1.2) the respective MyWELCOME applications (section 2.2.2) to enable the assessment exercises in the application (step 1.3). Once the TCNs solved the exercises (step 1.4) and the assessment score was computed (step 1.5), the MyWELCOME application sent them to the respective agents (step 1.6). Then the agents received the scores, saved them (step 1.7), and dispatched them to the TP (step 1.8) to conclude the Assessment phase.
2. In the Coordination phase, the first step was to activate/wake all the agents that

were supposed to participate in the coordination process (step 2.1). Then the TP sent a message to the agents (step 2.2) to start their coordination, which was executed in step 2.3. Once the agents coordinated and found solutions (step 2.3), one of them sent the solutions to the TP (step 2.4) such that the teacher could select one of them and submit it (step 2.5) as the final result. Additionally, agents did not inform the MyWELCOME applications at the end of the process as per the requirement of the user partner DIFE.

To look closely at the coordination of the agents, the steps of the second diagram (figure 4.3) are explained below:

Agents started their coordination when they received a signal from the TP (step 2). Since the profile information was necessary to compute the utility values according to the description in item 2 in section 4.2.1, all agents sent their profile information to the **dedicated agent** (step 3).

The dedicated agent should be one of the participant agents and the selection of the dedicated agent was done by the TP and specified in the configuration information that was sent to the agents so that all of them know the identity of the dedicated agent. After receiving the profile information from all participant agents, the dedicated agent generated the coalitions according to the description in item 1 in section 4.2.1 (step 4) and broadcasted these coalitions together with the profile information of all agents (step 5) such that the agents could compute their utility values according to the description in item 2 in section 4.2.1 (step 6) and send it back to the dedicated agent (step 7).

Once the dedicated agent collected all the utility values, it calculated the coalition and coalition structure values (step 8) before executing the CSGP solver algorithm, BOSS (described in item 2a in section 2.3.1) (step 9) to solve the LCC use-case.

To run the BOSS algorithm, it was necessary to use the SPARQL Construct query that is shown in figure 4.4. This query (in figure 4.4) constructed a *MACProblemInstance* (line 4) that should have the *number of agents* (line 5), the IDs of *participating agents* (line 5), the ID of the *use-case* (line 7), and the *coalitions* which did not violate the system constraints (line 8). Moreover, each of these coalitions (line 9) should have the IDs of the *member agents* (line 10) and a *value* (line 11) that was computed by the dedicated agent.

```
1. PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2. PREFIX mac: <http://localhost:8090/rdf4j/repositories/ajan_mac_ontology#>
3. CONSTRUCT {
4.     ?useCaseSubj    rdf:type    mac:MACProblemInstance ;
5.                     mac:hasNumberOfAgents    ?numOfAgents ;
6.                     mac:hasParticipants    ?pAgentId ;
7.                     mac:hasId    ?useCaseId ;
8.                     mac:hasFeasibleCoalitions    ?feasibleCoalitionNode .
9.    ?feasibleCoalitionNode    rdf:type    mac:CSGP-Coalition ;
10.                             mac:hasMembers    ?mAgentId ;
11.                             mac:hasValue    ?coalitionValue .
12. }
13. WHERE{
14.    ?useCaseSubj    rdf:type    mac:MACProblemInstance ;
15.                    mac:hasNumberOfAgents    ?numOfAgents ;
16.                    mac:hasParticipants    ?pAgentId ;
17.                    mac:hasId    ?useCaseId ;
18.                    mac:hasFeasibleCoalitions    ?feasibleCoalitionNode .
19.    ?feasibleCoalitionNode    rdf:type    mac:CSGP-Coalition ;
20.                              mac:hasMembers    ?mAgentId ;
21.                              mac:hasValue    ?coalitionValue .
22. }
```

Figure 4.4: SPARQL Construct query to execute BOSS algorithm as a BT node

Then the dedicated agent informed other agents (step 11) and TP (step 12) accordingly. At the end of the coordination process, all agents updated their status from "In_Coordination" to "Active" in Welcome Agent Repository (WAR) (section 2.2.2) through the Welcome Platform Manager (WPM) (section 2.2.2) (step 12). The final step of the LCC use-case was the submission of the final solution (step 13) that was selected from the proposed solutions by the teacher via TP.

The communication complexity of the LCC protocol in figure 4.3 was $6n$ where n was the number of agents that participated in the coordination process. If agents did not share the profile information through the dedicated agent (step 3 in 4.3) but shared it directly with each other ($n * (n - 1)$), the complexity of communication would increase to $n^2 + 3n$.

4.2.3 LCC Coordination Algorithms and Assumptions

This section presents the pseudocode of the LCC coordination algorithms. Moreover, the assumptions that are listed below were important for this use-case to work correctly.

Algorithm 1 is the first algorithm that agents execute when they start the LCC process. This algorithm includes actions for both the dedicated and participant agents. The participants send their profile information to the dedicated agent (line 2). The dedicated agent collects the profile information (lines 4-8), generates (line 9) and broadcasts the coalitions (line 10), computes its utility values (lines 11-13), collects the utility values of participants (lines 14-18), computes coalition values (line 19), executes the solver algorithm (line 20), stores (line 21) and broadcasts the results to participants (line 22) and Teacher Panel (TP) (line 23).

Algorithm 1 Start LCC Process

Input: Dataset A with n participant agents, $A = \{a_1, a_2, ..., a_n\}$ and dedicated agent ID, $A_{dedicated} \in A$
▷ The agent who runs this algorithm is called *this agent*, A_{this}
▷ $PI^{A_{this}}$ is the profile information of A_{this}

Result: Data structure $G = (grouping_1, grouping_2, ..., grouping_m)$ that represents the ranked list of LCC solutions where $grouping_x \in G$ is a better solution (i.e. has a higher suitability between members) than $grouping_y \in G$ when $x, y \in N$ and $x < y$. A solution, $grouping_x = \{group_1, group_2, ..., group_k\}$ is a set of groups. A group, $group_k = \{a_1, a_2, ..., a_t\}$ is a set of agents who are assigned to the same group.

1: **if** $A_{this} \neq A_{dedicated}$ **then**
2: send profile information, $PI^{A_{this}}$ to the dedicated agent, $A_{dedicated}$
3: **else if** $A_{this} = A_{dedicated}$ **then**
4: **while** $true$ **do**
5: **if** $n - 1$ agents returned their profile information or 45 seconds passed **then**
6: Terminate (**while loop**)
7: **end if**
8: **end while**
9: **Apply** the constraints 4.1, 4.2 and **generate coalitions** set, $C = \{c_1, c_2, .., c_m\}$
10: **Broadcast** the coalitions (C) and the profile information of all agents (PI^A) to all agents except the dedicated agent ($A/\{A_{dedicated}\}$)
11: **for each** $C^i \in \mathcal{C}$ where $A_{dedicated} \in C^i$ **do**
12: Compute utility values with equation 4.7, $uv(C^i, A_{dedicated})$
13: **end for**
14: **while** $true$ **do**
15: **if** $n - 1$ agents returned their utility values or 90 seconds passed **then**
16: Terminate (**while loop**)
17: **end if**
18: **end while**
19: **Compute** coalition values with equation 4.10, $cv(C^i)$.
20: **Run** *BOSS* algorithm to solve LCC and produce a rank list of grouping solutions, **G**
21: **Save** the solutions, **G**
22: **Broadcast** the solutions, **G**, to all agents except the dedicated agent ($A/\{A_{dedicated}\}$)
23: **Send** the solutions, **G**, to the TP
24: **end if**

The runtime complexity of algorithm 1 is $T^{pi} + T^{uv} + p * c^{dedicated}$ where T^{pi} and T^{uv} are the number of iterations to collect the profile information and utility values, respectively. Moreover, p and $c^{dedicated}$ are the number of LCC preferences and coalitions that contain $A_{dedicated}$.

Algorithm 2 is used by the participant agents for the computation of their utility values (lines 1-3).

Algorithm 2 The participants Compute Utility values

Input: A dataset C with m coalitions, $C = \{c_1, c_2, ..., c_m\}$ and the dedicated agent ID, $A_{dedicated}$

▷ The agent who runs this algorithm is called *this agent*, A_{this}

Result: A set, $\{(C^1, uv^1), (C^2, uv^2), ..., (C^i, uv^i)\}$ where (C^i, uv^i) represents the utility value, uv^i of the coalition, C^i

1: **for each** $C^i \in \mathcal{C}$ where $A_{this} \in C^i$ **do**
2: Compute utility values with equation 4.7, $uv(C^i, A_{this})$
3: **end for**
4: **Return** the utility values to the dedicated agent, $A_{dedicated}$

The runtime complexity of algorithm 2 is $p * c^{participant}$ where p and $c^{participant}$ are the number of LCC preferences and coalitions that contain the participant agent who computes the utility values.

LCC Coordination Assumptions

There were a few assumptions that needed to hold for the LCC use-case to work correctly:

Assumption 1. All agents run without any technical (e.g. server-side) errors.

Assumption 2. No agent can join the coordination after it was started.

Assumption 3. The modification of the configurations such as individual preferences, profile information, minimum, and maximum group sizes and so on during a coordination process will be taken into account in the next coordination process.

Assumption 4. The coordination cannot be stopped in the middle, either by the TCNs or the course instructor.

Assumption 5. If an agent does not return its profile information or utility values, then its preferences will not be considered in the computed solutions. Not returning such data does not break the process.

Assumption 6. If the dedicated agent fails, the LCC process will fail. It will need to be restarted again.

4.3 LCC Implementation

To implement the LCC use-case, the template BTs for the CSGP coordination protocol (figure 3.12) BTs were used. These template BTs covered most of the steps of the LCC protocol (section 4.2.2) since it was solved as a CSGP. Nevertheless, it was necessary to extend the template BTs with the LCC-related steps because those BTs were use-case independent. Moreover, the explanations of the CSGP coordination protocol BTs are provided in the MAJAN Guideline document and thus, the extensions that were specific to the LCC are discussed in this section without going into the details of those BTs. Lastly, the implementaion of the LCC use-case is provided in the *AkbarKazimov/LCC-CHC-Use-cases-with-MAJAN* [19] GitHub repository.

LCC Assessment phase:

The *Assessment phase* in LCC was not a part of the CSGP coordination protocol as described in figure 3.12. Therefore, two following BTs were implemented in addition to cover this phase:

1. **StartLccAssessmentBt** (figure 4.5) handled receiving the start message from the TP (step 1.1) and dispatching it to the MyWELCOME application (step 1.2).

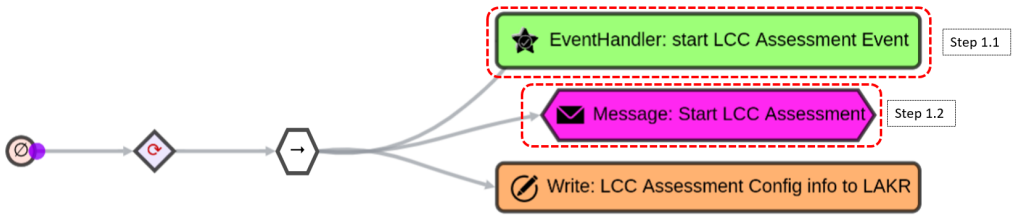

Figure 4.5: Start LCC Assessment phase BT

2. **LccAssessmentCompleteBt** (figure 4.6) was used to receive the assessment scores from the MyWELCOME application (step 1.6), save them (step 1.7), and dispatch them to the TP (step 1.8).

[19] `https://github.com/AkbarKazimov/LCC-CHC-Use-cases-with-MAJAN/tree/main/LCC`

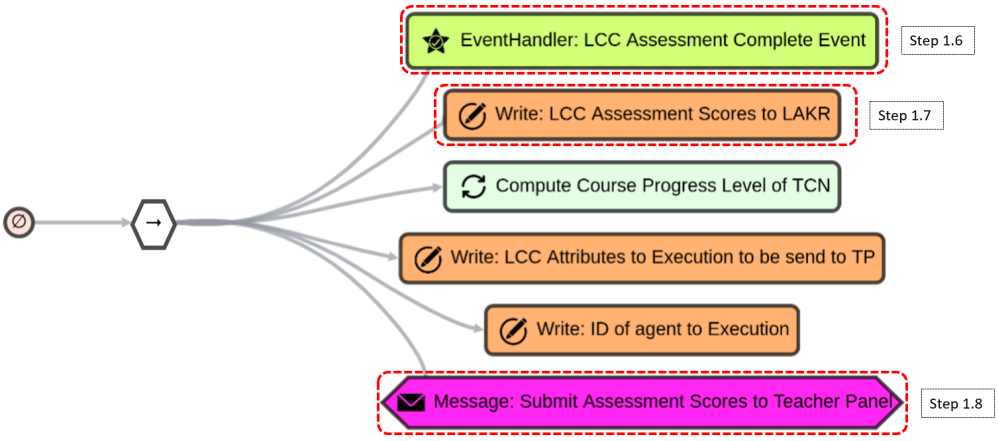

Figure 4.6: Complete LCC Assessment phase BT

LCC Coordination phase:

The extensions that were made to the CSGP coordination protocol BTs to solve the LCC use-case (figure 4.3) are described below.

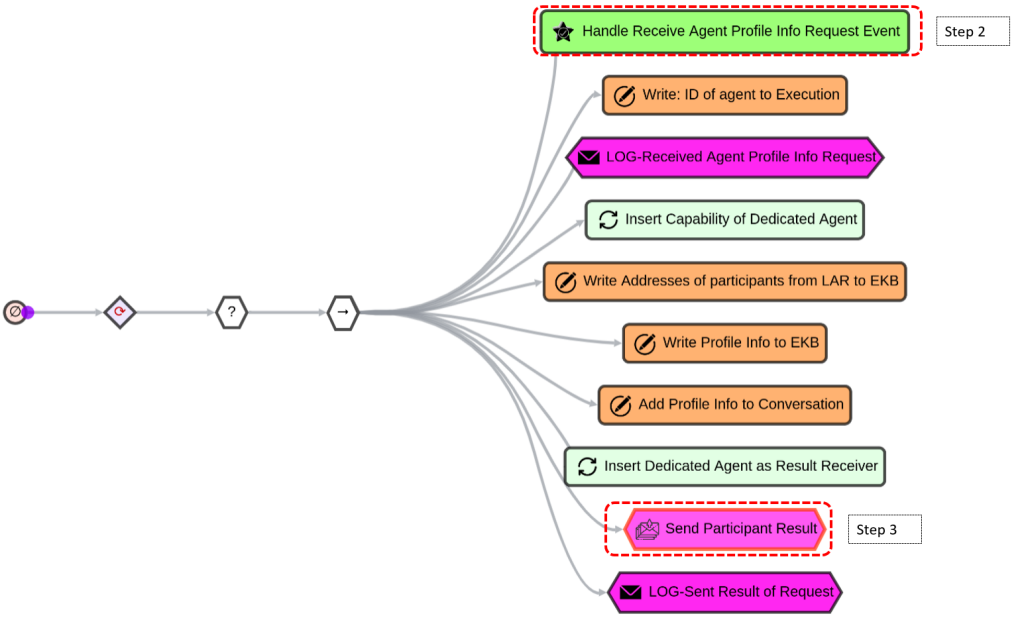

Figure 4.7: The initiating BT of the LCC protocol

Steps 2 and 3 of the LCC protocol (in figure 4.3) were implemented in the BT in figure 4.7.

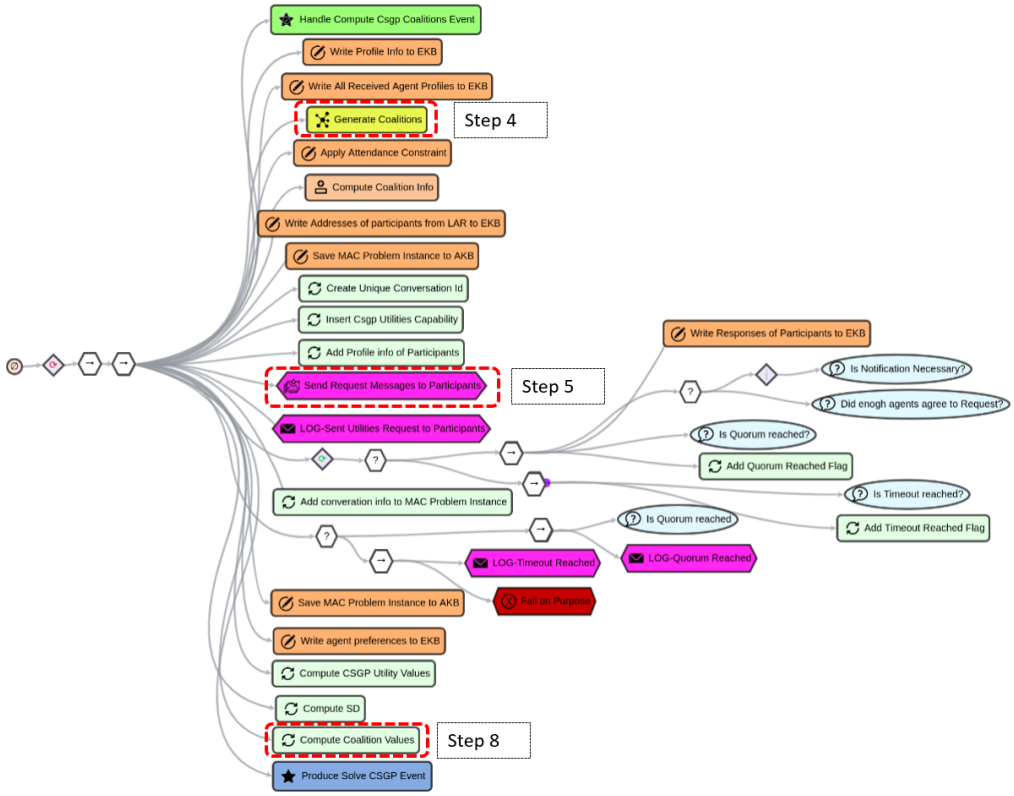

Figure 4.8: Compute CSGP Coalitions Behavior Tree

```
1. PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2. PREFIX welcome: <https://raw.githubusercontent.com/gtzionis/WelcomeOntolo
gy/main/welcome.ttl#>
3. PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
4. PREFIX mac: <http://localhost:8090/rdf4j/repositories/ajan_mac_ontology#>
5. DELETE {
6.     ?bnode     mac:hasFeasibleCoalitions     ?feasibleCoalitionNode .
7.     ?feasibleCoalitionNode     rdf:type     mac:CSGP-Coalition ;
8.                                ?p     ?o.
9. }
10. WHERE{
11.     ?bnode     rdf:type     welcome:LCCUseCase ;
12.                welcome:hasLessonId     ?lessonId ;
13.                welcome:hasAbsenceAllowance     ?allowedAttendanceLimit ;
14.                mac:hasFeasibleCoalitions     ?feasibleCoalitionNode .
15.     ?feasibleCoalitionNode     rdf:type     mac:CSGP-Coalition ;
16.                                mac:hasSize     ?groupSize ;
17.                                welcome:hasTotalAttendanceAmount
?attendedAmount;
18.                                ?p  ?o.
19.     FILTER((xsd:integer(?lessonId) != 1) && ((xsd:integer(?attendedAmount)
+ xsd:integer(?allowedAttendanceLimit)) < xsd:integer(?groupSize)))
20. }
```

Figure 4.9: SPARQL Update query to apply attendance constraint (2d) of LCC

The BT that was called *Compute CSGP Coalitions BT* (figure 4.8) covered **steps 4, 5, and 8** of the LCC protocol. Moreover, the attendance constraint (item 2d in section 2.2.3) of LCC was applied in this BT. To do so, the **Update** query in figure 4.9 was used which was executed after generating the coalitions, and it removed all the coalitions that violate the attendance constraint of LCC. This query retrieved the number of students who were allowed to be absent in the previous lesson (**line 13**), the size of the coalition (**line 16**), and the number of students in the coalition who were present in the previous lesson (**line 17**). **Line 19** filtered the coalitions in which more students missed the previous lesson than the allowed limit. Finally, the filtered coalitions were removed in the *DELETE* clause (**lines 5-9**).

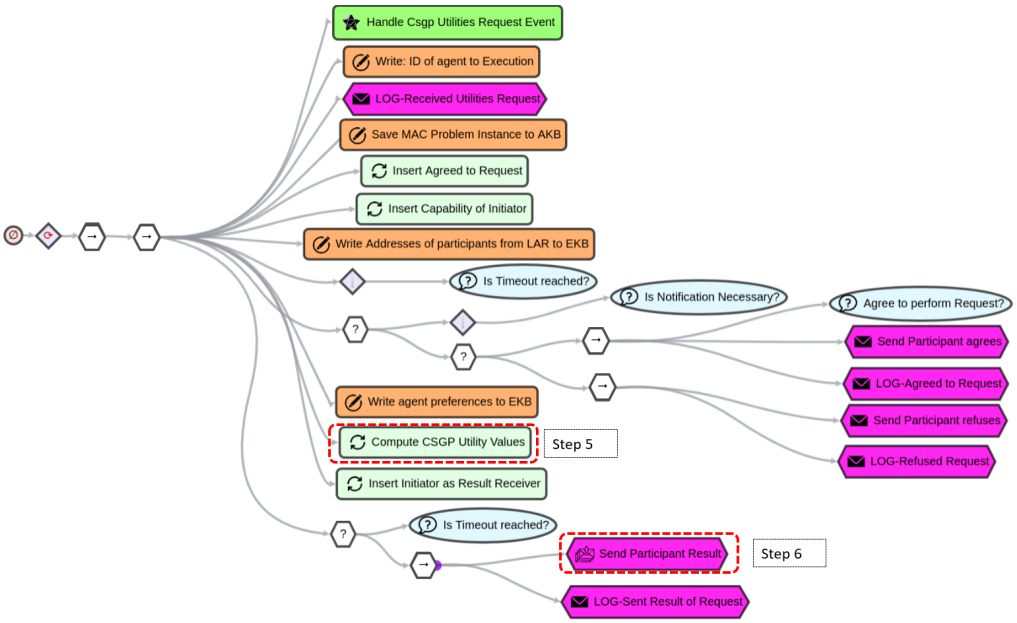

Figure 4.10: Handle CSGP Utilities BT

```
1. PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2. PREFIX ajan: <http://www.ajan.de/ajan-ns#>
3. PREFIX welcome: <https://raw.githubusercontent.com/gtzionis/WelcomeOntol
ogy/main/welcome.ttl#>
4. PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
5. PREFIX mac: <http://localhost:8090/rdf4j/repositories/ajan_mac_ontology#>
6. CONSTRUCT {
7.     ?bnode     rdf:type     welcome:LCCUseCase ;
8.                mac:hasId     ?lccId ;
9.                mac:hasFeasibleCoalitions     ?feasibleCoalitionNode .
10.    ?feasibleCoalitionNode     mac:hasValue     ?uValueBnode .
11.    ?uValueBnode     rdf:type     mac:UtilityValue ;
12.                     mac:isComputedBy     ?agentId ;
13.                     mac:hasValue     ?ttlV .
14. }
15. WHERE {
16.     {
17.     SELECT DISTINCT ?feasibleCoalitionNode ?agentId ?bnode ?lccId
(SUM(?uValue) AS ?ttlV)
18.     WHERE {
19.         # get This agent
20.         ?agent     rdf:type     ajan:Agent, ajan:ThisAgent ;
21.                    ajan:agentId     ?agentId .
22.         # get Feasible Coalitions which are sent by Dedicated agent
23.         ?bnode     rdf:type     welcome:LCCUseCase ;
24.                    mac:hasId     ?lccId ;
25.                    mac:hasFeasibleCoalitions     ?feasibleCoalitionNode .
26.         # make sure that the agent (i.e. this.agent) who is computing the
utility value is included in the coalition
27.         FILTER EXISTS { ?feasibleCoalitionNode  mac:hasMembers
?agentId. }
28.         # get the coalition information
29.         ?feasibleCoalitionNode     rdf:type     mac:CSGP-Coalition ;
30.                                    mac:hasCommonGender     ?commonGender ;
31.                                    mac:hasCommonNation     ?commonNation .
32.         # get preferences of this.agent
33.         ?agent     welcome:hasLCCPreferencesInfo     ?aLccPrfInfo .
34.         ?aLccPrfInfo     rdf:type     welcome:LCCPreferencesInfo ;
35.                          welcome:hasLccGenderPreference     ?aGenderPref ;
36.                          welcome:hasLccNationPreference     ?aNationPref ;
37.                          welcome:hasLccGenderPrefWeight     ?gWeight ;
38.                          welcome:hasLccNationPrefWeight     ?nWeight .
39.         # get gender and nation of this.agent
40.         ?agent     welcome:hasLCCPersonalInfo     ?aLccPsnlInfo .
41.         ?aLccPsnlInfo     rdf:type     welcome:LCCPersonalInfo ;
42.                           mac:hasId     ?lccId ;
43.                           welcome:hasGender     ?aGender ;
44.                           welcome:hasNationality     ?aNation .
45.         BIND(IF(LCASE(?aGenderPref) = "dont␣mind" || (LCASE(?aGenderPref)
              = "same" && ?aGender = ?commonGender) || (LCASE(?aGenderPref) =
              "mixed" && LCASE(?commonGender) = "mixed"), xsd:float(?gWeight),
              -xsd:float(?gWeight)) AS ?genderUValue)
46.
47.             BIND(IF(LCASE(?aNationPref) = "dont␣mind" || (LCASE(?aNationPref)
              = "same" && ?aNation = ?commonNation) || (LCASE(?aNationPref) =
              "mixed" && LCASE(?commonNation) = "mixed"), xsd:float(?nWeight),
              -xsd:float(?nWeight)) AS ?nationUValue)
48.             BIND((?genderUValue + ?nationUValue) AS ?uValue)
49.             } GROUP BY ?feasibleCoalitionNode ?agentId ?bnode ?lccId
50.         }
51.     BIND(BNODE() AS ?uValueBnode)
52. }
```

Figure 4.11: SPARQL Construct query to apply individual preferences when computing the utility values for LCC

The CSGP utility values were computed (steps 6 and 7) in *Handle CSGP Utilities BT* (figure 4.10). Additionally, the query in figure 4.11 was used to apply the individual LCC preferences when computing the utility values. In this query, **lines 19-21** and **lines 22-25** retrieved the ID of the agent who was computing utility values and the coalitions that were required to compute utility values, respectively. **Lines 26-27** ensured that the agent computed the utility values only for the coalitions that it was a member of. **Lines 28-44** fetched the coalition, the preferences, and the profile information of the agent. **Lines 45 and 47** checked whether the gender and nationality preferences of the agent were satisfied or violated, respectively. Then the individual satisfaction or violation values of gender and nationality preferences were summed in **line 48** to compute the utility value for one coalition. This process was executed iteratively (**lines 16-50**) to compute separate utility values for each coalition.

```
1. PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2. PREFIX ajan: <http://www.ajan.de/ajan-ns#>
3. PREFIX welcome: <https://raw.githubusercontent.com/gtzionis/WelcomeOntolo
gy/main/welcome.ttl#>
4. PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
5. PREFIX mac: <http://localhost:8090/rdf4j/repositories/ajan_mac_ontology#>
6. INSERT {
7.     ?feasibleCoalition    mac:hasValue    ?coalitionValue .
8. }
9. WHERE{
10.     {
11.     SELECT ?feasibleCoalition ?sdValue ?sdWeight (SUM(?utilityValue)
12.     AS ?sumOfUtilities)
13.     WHERE {
14.         ?lccUseCase    rdf:type    welcome:LCCUseCase ;
15.                        mac:hasStatus    ?lccStatus ;
16.                        mac:hasId    ?lccId ;
17.                        welcome:hasSdWeight    ?sdWeight ;
18.                        mac:hasFeasibleCoalitions    ?feasibleCoalition .
19.         FILTER(LCASE(?lccStatus) = "running")
20.         ?feasibleCoalition    rdf:type    mac:CSGP-Coalition .
21.         OPTIONAL {
22.             ?feasibleCoalition    mac:hasValue    ?utilityValueNode .
23.             ?utilityValueNode    rdf:type    mac:UtilityValue ;
24.                                  mac:hasValue    ?utilityValue .
25.         }
26.         OPTIONAL {
27.             ?feasibleCoalition    welcome:hasStandardDeviation ?sdValue .
28.         }
29. } GROUP BY ?feasibleCoalition ?sdValue    ?sdWeight
30. }
31. BIND(IF(BOUND(?sumOfUtilities), ?sumOfUtilities, 0) AS ?boundSumOfUtils)
32. BIND(xsd:float(?sdValue) AS ?fSdValue)
33. BIND(xsd:float(?sdWeight) AS ?fSdWeight)
34. BIND(xsd:float(?boundSumOfUtils) AS ?fBoundSumOfUtils)
35. BIND(((1-?fSdWeight) * ?fBoundSumOfUtils) - (?fSdWeight * ?fSdValue) AS
    ?coalitionValue)
36. }
```

Figure 4.12: SPARQL Update query to compute coalition values for LCC

Since it was necessary to take into account the SD of the coalitions as stated in item

3 in section 4.2.1 when computing the coalition values in LCC, the SPARQL query in figure 4.12 was formulated and used. This query retrieved the total utility value (**lines 22-24**) and SD value (**line 27**) of each coalition to compute the coalition value in **line 35** by also taking into account the importance weight of the SD.

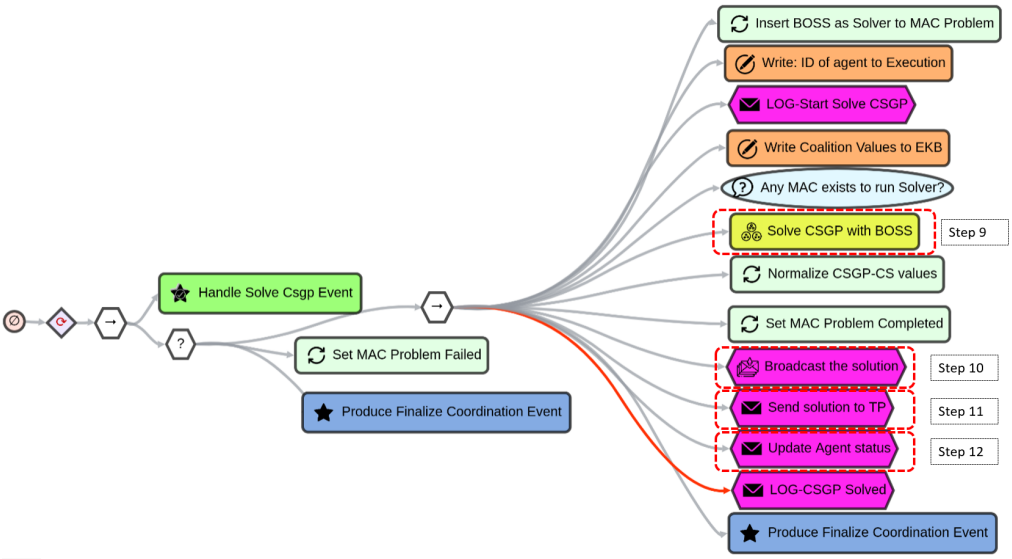

Figure 4.13: Solve CSGP BT

Steps 9, 10, 11, and 12 of the LCC protocol (figure 4.3) were covered in *Solve CSGP BT* which is described in figure 4.13. In order to send the solutions of LCC to the TP (**step 11**) and to update the status of agents (**step 12**), two message nodes were added to the CSGP coordination protocol BTs since these steps were specific to the LCC use-case.

Moreover, there was no need to modify any query of the CSGP BTs to apply the group size constraint (item 2b in section 2.2.3) of the LCC use-case, since the template BTs were able to accept the *minimum* and *maximum coalition sizes*.

LCC feature in the MyWELCOME application and Teacher Panel

The MyWELCOME application and the Teacher Panel (TP) implemented the necessary functionalities to deliver the LCC feature successfully. The steps for the course instructor and the TCNs are described below with the respective UI pictures of MyWELCOME and TP.

In the TP, the course instructor firstly selects a classroom, a lesson ID, and a date to start the assessment phase of LCC as shown in figure 4.14.

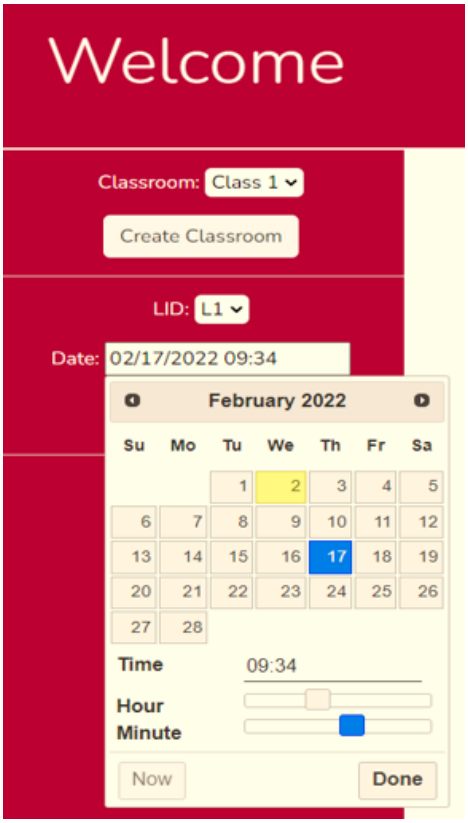

Figure 4.14: Start Assessment feature in TP

The TCNs receive the assessment notification in the MyWELCOME application as shown in figure 4.15.

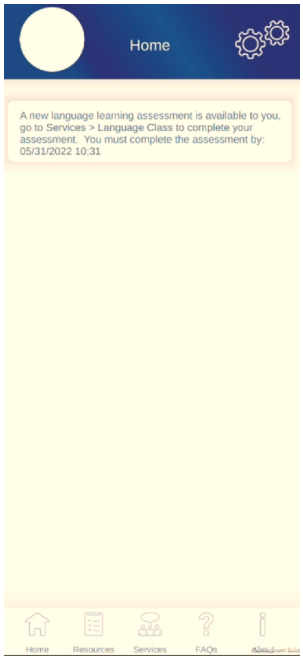

Figure 4.15: Assessment Notification in the MyWELCOME application

Then, the MyWELCOME application displays the assessment exercise types (figure 4.16) and the selected assessment questions (figure 4.17).

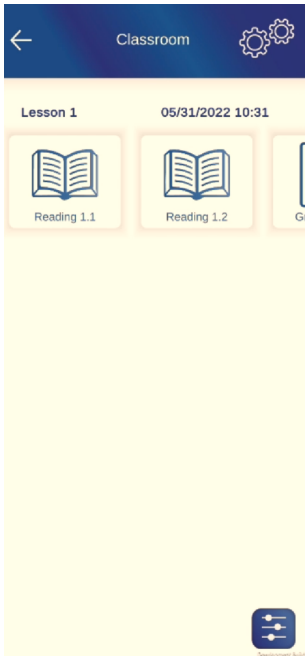

Figure 4.16: Assessment exercise types in the MyWELCOME application

Figure 4.17: Assessment questions in the MyWELCOME application

After the TCNs complete the assessment exercises, the TP receives the respective scores and displays them as shown in figure 4.18.

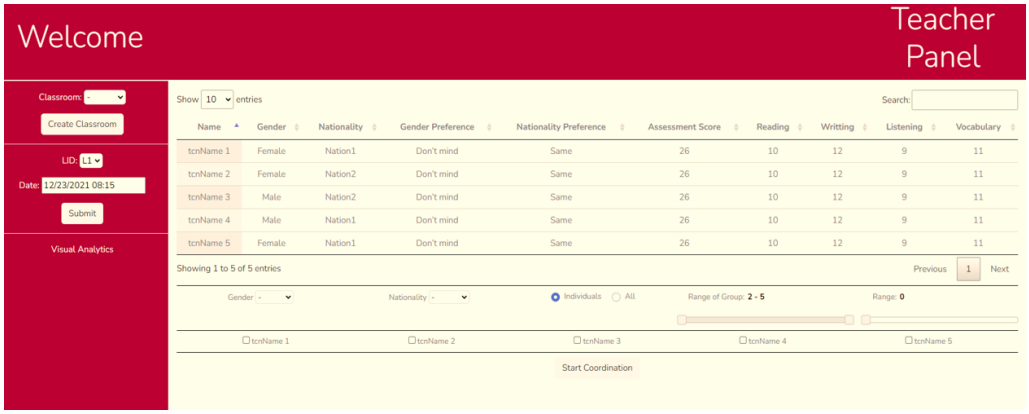

Figure 4.18: Start coordination feature in TP

TCNs assign their preferences in the MyWELCOME application via the UI in figure 4.19.

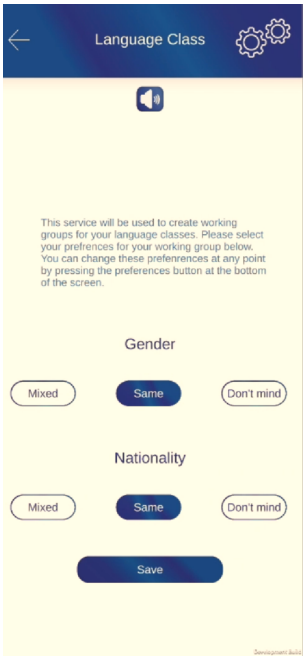

Figure 4.19: Assign preferences in the MyWELCOME application

To start the coordination of agents, the course instructor selects the TCNs who will participate in the coordination and clicks on the *Start Coordination* button (figure 4.18). The agents send the results of the coordination to the TP after they complete their coordination. The results are visualized as shown in figure 4.20.

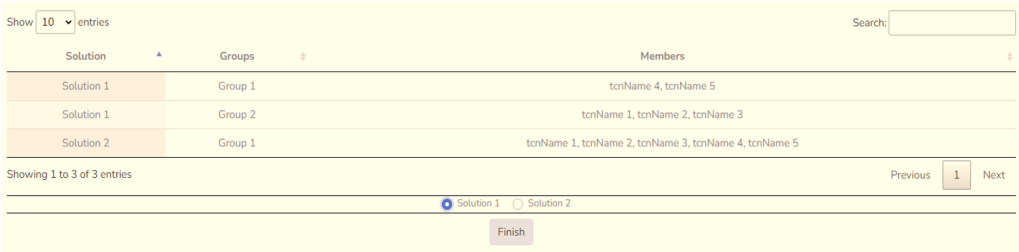

Figure 4.20: Visualization of the solutions in TP

The last step for the course instructor is to select one of the solutions and submit it as the final solution by clicking on the *Finish* button (figure 4.20).

4.4 LCC Evaluation

The LCC use-case was provided by the user partner, DIFE [20] to be developed as part of the MyWELCOME application. This section presents its evaluation results. It is important to note that, this use-case had not been implemented before, and therefore, its implementation in this thesis was the first where the process of dividing students into working groups was automatized. The creation of working groups had been manually done by the course instructor so far. Therefore, the user partner did not have any supervised data which could be used as the ground truth to compare the results of this work. Thus, the first step was to create various realistic TCN profiles and then collect the confirmation of the user partner about the profiles. Afterward, the LCC process was simulated such that the agents could compute grouping solutions. To evaluate the solutions, the Customer Satisfaction Score (CSAT) [53] was used, which is a metric to measure the satisfaction of users on a Likert scale as shown in table 4.1.

Likert score	Adjective
1	Very Dissatisfied
2	Dissatisfied
3	Neutral
4	Satisfied
5	Very Satisfied

Table 4.1: Likert scale [22, 45]

The subsections present the dataset, configurations, and evaluation results.

4.4.1 Settings

Eleven data records, each of which consisted of 10, 15, or 20 TCN profile information, were created to be used as the dataset. A TCN profile information included the values for *gender*, *nationality*, *Course Progress Level (CPL)*, *attendance info to the previous lesson*, *gender preference*, and *nationality preference* attributes. Moreover, each data record had its characteristics. Table 4.2 describes the first data record in which the TCNs were balanced in terms of their gender and nationality attributes (4 male vs 6 female). They were more inclined to prefer mixed groups, and only 20% of TCNs missed the previous lesson. Females mostly preferred the same gender and the mixed nationality groups, and males mostly preferred the mixed gender and the same nationality groups.

[20] `https://web.gencat.cat/ca/inici`

Agent ID	Gender	Nationality	CPL	Attendance	Gender Preference	Nationality Preference
1	Male	A	72	No	Don't mind	Same
2	Male	A	84	Yes	Mixed	Same
3	Female	B	73	Yes	Same	Mixed
4	Male	C	53	Yes	Mixed	Mixed
5	Female	D	25	Yes	Don't mind	Mixed
6	Female	B	44	Yes	Same	Mixed
7	Female	C	89	Yes	Mixed	Don't mind
8	Female	C	57	Yes	Mixed	Mixed
9	Female	D	95	Yes	Same	Mixed
10	Male	C	60	No	Mixed	Same

Table 4.2: LCC data record 1

The rest (i.e. 10) of the data records are provided in appendix C.

Additionally, four configurations were given by the user partner for the LCC system constraints, since they were assigned dynamically by the course instructor in practice. These configurations are listed below:

1. Configuration 1:
   (a) **Group Size**: minimum 2 maximum 5 TCNs in each group
   (b) **Attendance**: maximum 1 TCN in each group who missed the previous lesson
   (c) **CPL vs Satisfaction of preferences importance weights**: 0.75 vs 0.25
   (d) **Gender vs Nationality preferences importance weights:** 0.75 vs 0.25
2. Configuration 2:
   (a) **Group Size**: minimum 2 maximum 5 TCNs in each group
   (b) **Attendance**: maximum 1 TCN in each group who missed the previous lesson
   (c) **CPL vs Satisfaction of preferences importance weights**: 0.1 vs 0.9
   (d) **Gender vs Nationality preferences importance weights:** 0.75 vs 0.25
3. Configuration 3:
   (a) **Group Size**: minimum 2 maximum 3 TCNs in each group
   (b) **Attendance**: maximum 1 TCN in each group who missed the previous lesson
   (c) **CPL vs Satisfaction of preferences importance weights**: 0.75 vs 0.25
   (d) **Gender vs Nationality preferences importance weights:** 0.5 vs 0.5
4. Configuration 4:
   (a) **Group Size**: minimum 3 maximum 4 TCNs in each group
   (b) **Attendance**: maximum 1 TCN in each group who missed the previous lesson
   (c) **CPL vs Satisfaction of preferences importance weights**: 0.25 vs 0.75
   (d) **Gender vs Nationality preferences importance weights:** 0.25 vs 0.75

4.4.2 Results

The agents computed 44 results by running each data record (i.e. 11 data records) with each configuration (i.e. 4 configurations). All the results were provided and evaluated separately on the Likert scale (table 4.1) by two users from DIFE. To compute the final evaluation score, the arithmetic mean of the evaluation scores of 44 results was calculated. The final evaluation score was **5.0** which showed that the users were *very satisfied* with the implementation and results of the LCC use-case. The results of the first data record that was executed with all the configurations are provided below. The rest of the results (i.e. 40 results) are provided in appendix C.

Results of Data Record 1: Each solution is a grouping that consists of groups. The numbers in the solutions represent agent IDs. Only the 3 best solutions that are ranked from the best to the worst are listed.

1. **Configuration 1**
   - **Total Computation time**: 141 seconds
   - **Solution List**:
     (a) Rank 1: [[4, 8, 10], [5, 6], [2, 7, 9], [1, 3]]
     (b) Rank 2: [[4, 5, 6, 8, 10], [2, 7, 9], [1, 3]]
     (c) Rank 3: [[2, 7, 9], [4, 5, 6], [1, 3], [8, 10]]
   - **Customer Satisfaction Score**: 5
2. **Configuration 2**
   - **Total Computation time**: 138 seconds
   - **Solution List**:
     (a) Rank 1: [[4, 8, 10], [5, 6], [1, 2, 7], [3, 9]]
     (b) Rank 2: [[1, 2, 4, 7], [3, 9], [5, 6], [8, 10]]
     (c) Rank 3: [[2, 4, 8, 10], [1, 7], [3, 9], [5, 6]]
   - **Customer Satisfaction Score**: 5
3. **Configuration 3**
   - **Total Computation time**: 132 seconds
   - **Solution List**:
     (a) Rank 1: [[2, 7, 9], [4, 8, 10], [1, 3], [5, 6]]
     (b) Rank 2: [[2, 7, 9], [4, 5, 6], [1, 3], [8, 10]]
     (c) Rank 3: [[1, 2, 3], [4, 8, 10], [5, 6], [7, 9]]
   - **Customer Satisfaction Score**: 5
4. **Configuration 4**
   - **Total Computation time**: 143 seconds
   - **Solution List**:
     (a) Rank 1: [[4, 5, 6, 10], [1, 3, 8], [2, 7, 9]]

(b) Rank 2: [[1, 2, 7, 9], [4, 5, 6], [3, 8, 10]]

(c) Rank 3: [[5, 6, 8, 10], [1, 3, 4], [2, 7, 9]]

- **Customer Satisfaction Score**: 5

Chapter 5
Co-Habitation Coordination Use-Case

The description, implementation, and evaluation of the CHC use-case (section 2.2.3) of the MyWELCOME application (section 2.2.2) are presented in this chapter.

5.1 CHC Description

In the Co-Habitation Coordination (CHC) use-case, agents were responsible to form the groups of TCNs in which the members of the same groups could best match with each other based on their individual preferences and the system constraints (section 2.2.3) to share an apartment. After agents coordinated themselves into groups, they needed to inform the TCNs through the MyWELCOME application by sharing the name and contact information of other TCNs who were assigned to the same groups. Since the contact information should be shared with other TCNs, their consent to participate in the CHC use-case should be asked in the MyWELCOME application so that their agents can find potential roommates for them by coordinating with other agents and by satisfying as many CHC constraints as possible.

It is important to note that, agents were not responsible to search for available apartments or matching the TCNs with landlords. Such activities could be performed by the TCNs.

Individual Preference	Importance Weight
1. Age	10 (the most important)
2. Gender	9
3. Family	5
4. Nationality	8
5. Religion	6
6. Ethnicity	7
7. Location of Apartment	2
8. Accessibility for Disabled	3
9. Rental Period	1 (the least important)
10. Share with	4

Table 5.1: Default importance weights of CHC individual preferences

Section 2.2.3 provided the complete list of CHC individual preferences and the system constraints. It should be pointed out that each individual preference had an importance weight that described how important it was compared to other individual preferences, and these weights could be set by the TCNs through the MyWELCOME application. Table 5.1 presents the default importance weights that were provided by the user partner, PRAKSIS [21] who provided the CHC use-case:

5.2 CHC Conceptual Solution

This section begins with detailed information about the grouping algorithms that can solve the CHC use-case. Then, the CHC coordination protocol and its implementation are discussed without going into detail. Finally, the evaluation of the CHC implementation is presented.

5.2.1 Analysis of Grouping Algorithms for CHC

The list of the requirements that should be supported by the grouping algorithms so that they can solve the CHC use-case is given below:

1. CHC solutions should be sets of groups based on the description in section 2.2.3.
2. CHC constraints should be taken into account when computing the solutions.
3. In CHC, the minimum and the maximum group sizes were unknown. Thus, it was not possible to determine the number of groups to be produced in a CHC solution.
4. The solution of CHC should be optimal or deterministic.
5. In CHC, there could be hundreds of TCNs. Therefore, the algorithm should be able to compute a solution for at least a few hundred agents.

The grouping algorithms that were presented in section 2.3.1 are analyzed to identify the one(s) that can solve CHC. The comparison of the algorithms for CHC is described in figure 5.1.

[21] `https://praksis.gr/`

Algorithm	Outputs set of groups	Supports constraints	Requires the amount of groups	Optimal/ Determenistic	Supports (at least) 100 agents	Other
K-Means (McQueen, 1967)	Yes	No	Yes	Yes	Yes	-
COP-KMeans (Wagstaff et al, 2001)	Yes	Yes	Yes	Yes	Yes	No solution if a constraint is violated
Hierarchical Clustering (Nielsen, 2016)	Yes	No	Yes	Yes	Yes	-
DBSCAN (Ester et al, 1996)	Yes	No	No	Yes	Yes	-
ACE (Alqurashi et al, 2019)	Yes	No	No	Yes	Yes	No available implementation
Recommender Systems (Potts, et al., 2018), (Palomares et al, 2021)	No	-	-	-	-	-
BCOP-KMeans (Okabe et al, 2018)	Yes	Yes	Yes	Yes	Yes	Implementation in Matlab
CoExDBSCAN (Ertl et al, 2020)	Yes	Yes	No	Yes	Yes	No solution if a constraint is violated
ACS (Taguelmimt et al, 2021)	Yes	Yes	No	No	Yes	-
BOSS (Aknine et al, 2021)	Yes	Yes	No	Yes	No	Implementation in JAVA
HDBSCAN* (Campello R. J. et al, 2015)	Yes	Yes	No	Yes	Yes	Implementation in JAVA

Figure 5.1: Comparison of Grouping algorithms for CHC

The **K-Means** (item 1a in section 2.3.1), **COP-KMeans** (item 1b in section 2.3.1), **Hierarchical Clustering** (item 1g in section 2.3.1), and **BCOP-KMeans** (item 1c in section 2.3.1) algorithms were ruled out as they all require the number of clusters, k, to be predefined which violated requirement 3 given above.

Moreover, the **DBSCAN** (item 1d in section 2.3.1) and **ACE** (item 1h in section 2.3.1) algorithms could not solve the CHC use-case because they do not support the constraints which was necessary according to requirement 2. Although the **CoExDBSCAN** (item

1e in section 2.3.1) algorithm supports constraints, it does not find any solution if any constraint is violated.

Additionally, the **Recommender Systems** (item 3 in section 2.3.1) do not compute a grouping solution that violated requirement 1.

The **ACS** (item 2b in section 2.3.1) algorithm could solve CHC but it does not guarantee to find the optimal solution (requirement 4). Although the **BOSS** (item 2a in section 2.3.1) algorithm finds the optimal solution, it cannot work for more than 27 agents, which violated requirement 5.

Finally, the **HDBSCAN*** (item 1f in section 2.3.1) algorithm met all the requirements that are listed above. Therefore, it was selected to solve the CHC use-case. Alternatively, the ACS algorithm could also solve it if a non-guaranteed solution would be acceptable.

Projection of CHC to Clustering:

Since the HDBSCAN* solves the clustering problems, the CHC problem was reduced to a clustering problem in this paragraph. In this regard, a group of TCNs in CHC corresponded to a cluster in a clustering problem. As stated in section 2.3.1, the objective of the clustering problem was to group objects in a way that the objects in the same clusters were more similar to each other than the objects in other clusters. And in CHC, agents coordinated themselves into groups in which the dissimilarities (i.e. distance) between the agents in the same groups were minimized (equation 5.9).

To solve CHC as a clustering problem, it was necessary to compute the distance scores (equation 5.4) between the agents to be passed as the input to the clustering solver algorithms (in this case, HDBSCAN*). Based on this input, the solver algorithms find the clusters in which the objects are similar to each other. To compute a distance score, agents compared the individual preferences of their TCNs with the profile information of other TCNs.

$$A = \{a_1, a_2, ..., a_n\} \text{ is the set of agents} \tag{5.1}$$

$$P^{A_i} \text{ is the set of individual preferences of agent } i \tag{5.2}$$

$$W^{A_i} \text{ is the set of weights of the individual preferences of agent } i \tag{5.3}$$

$$d(A_i, A_j) = \sum_{m=1}^{|P^{A_i}|} (p(P_m^{A_i}, A_j) * W_m^{A_i}) \tag{5.4}$$

$$p(P_m^{A_i}, A_j) = \begin{cases} 0.001, & \text{if } j\text{th agent satisfies } m\text{th preference of } i\text{th agent} \\ 2, & \text{Otherwise} \end{cases} \tag{5.5}$$

where $p(P_m^{A_i}, A_j)$ function calculates whether the preference of an agent is satisfied by another agent and $d(A_i, A_j)$ is the distance function that computes the distance between two agents.

The distance function (equation 5.4) ensured that the distance was minimum between two agents if their preferences were satisfied (i.e. they were suitable for each other). Since each agent computed a distance score from its perspective (i.e. based on its individual preferences), there were two different distance scores between two agents, each of which was calculated by one of the agents. The clustering solver algorithms expect only one distance score between two data points (i.e. agents). To overcome this issue, this work referred to Reciprocal Recommender System (RRS) (section 3) since the RRS matches the users with each other based on their preferences, and this technique fit the CHC use-case.

To compute a single score (also called the *reciprocal score*) between two users, Pizzato et al. [61] computed the **Harmonic mean** of the scores in their work. The function $r(A_i, A_j)$ to compute the reciprocal scores for CHC is described in equation 5.6.

$$r(A_i, A_j) = \frac{2}{\frac{1}{d(A_i,A_j)} + \frac{1}{d(A_j,A_i)}} \tag{5.6}$$

Moreover, to consider the system constraint of CHC (section 2.2.3) that required the members of the same groups to speak a common language, the necessary constraint data was computed (equation 5.7).

$$l(A_i, A_j) = \begin{cases} 1, & \text{if agents } i \text{ and } j \text{ can speak a common language} \\ 0, & \text{Otherwise} \end{cases} \tag{5.7}$$

$$g(A_i, A_j) = \begin{cases} 1, & \text{if agents } i \text{ and } j \text{ are in the same group} \\ 0, & \text{Otherwise} \end{cases} \tag{5.8}$$

The objective of CHC was to minimize the sum of the reciprocal scores (equation 5.9) to find the best grouping solution in which the agents were most suitable to each other and the system constraint was satisfied.

$$\begin{aligned} \text{Minimize } f = \sum_{i=1}^{|A|} \sum_{j=i+1}^{|A|} r(A_i, A_j) \\ \text{subject to } g(A_i, A_j) = 1 \\ l(A_i, A_j) = 1 \end{aligned} \tag{5.9}$$

Once the distance scores and the constraints were ready, agents had all the required input data to solve the use-case as a clustering problem.

5.2.2 CHC Coordination Protocol

The template BTs for the clustering coordination protocol (figure 3.13) that were provided in the MAJAN Guideline document were extended to design the coordination protocol (figure 5.2) of the CHC use-case. Since the explanations of the clustering coordination protocol BTs were provided in the guideline document, the focus is on the CHC-related extensions in this section.

As shown in figure 5.2, the coordination process was initiated by the TCNs through the MyWELCOME application (**step 1**). The agent, whose TCN started the coordination process, became the dedicated agent. Each TCN was limited to running only one CHC process at a time because otherwise, hundreds of TCNs could start hundreds of processes in parallel, causing a crash in the server. **Step 2** ensured that the dedicated agent runs only one CHC process. In **steps 3 and 6**, the agents updated their status such that they appear "In_Coordination" in the system. **Steps 4, 5, 7, 8, 9, 10, 11, and 12** of the CHC protocol were implemented in the clustering coordination protocol as shown in figure 3.13. The final step (i.e. **step 13**) was to inform the TCNs about the grouping solution of the coordination process. In this step, each agent informed its TCN about the contact information of other TCNs who were assigned to the same group.

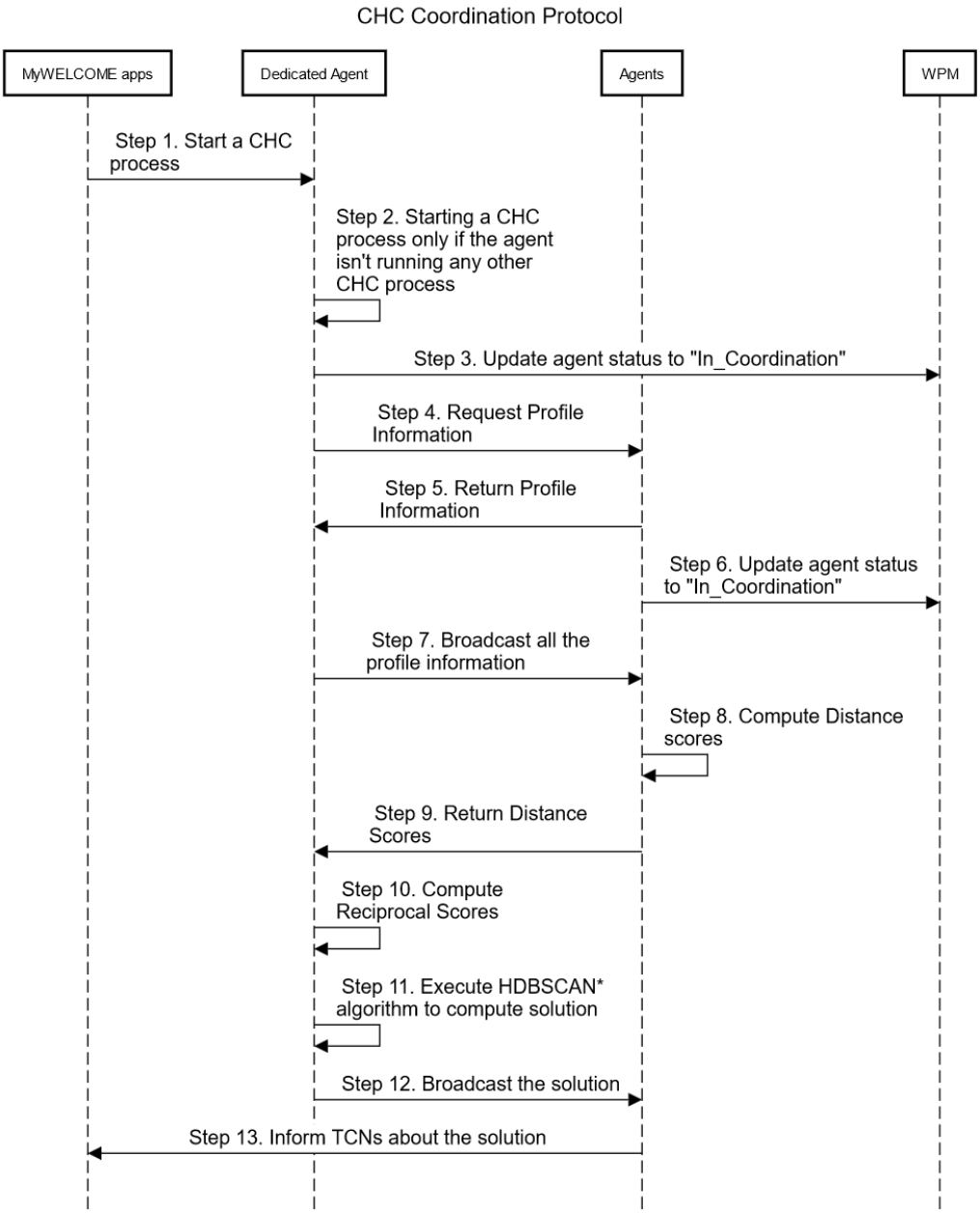

Figure 5.2: CHC Coordination Protocol

The complexity of the communication of the CHC coordination protocol was $7n$ where n was the number of agents that participated in the process.

5.2.3 CHC Coordination Algorithms and Assumptions

The assumptions that were made in the CHC use-case and the pseudocode of the CHC coordination algorithms are described below.

Algorithm 3 describes the steps for an agent to be able to start and execute a CHC process successfully. Since each TCN is allowed to execute only one CHC process at a time, line 1 checks whether the agent already runs a CHC process or not. The agent whose TCN starts the CHC becomes the dedicated agent. Then, the agent requests (line 2) and waits for other participant agents to return their profile information (lines 3-7). After either the profile information is returned or the specified amount of time passes, the dedicated agent broadcasts (line 8) the complete profile information and computes its distance scores (line 9). After the distance scores of the participant agents are collected (lines 10-14), the dedicated agent computes the reciprocal scores (line 15), executes the **HDBSCAN** algorithm (line 16), and saves the grouping solution of the coordination (line 17). Finally, the agent sends the solution to its TCN (line 19) and other participants (line 18).

Algorithm 3 Start CHC Process

Input: A Dataset A with n non-inactive (i.e. active or in_coordination) agents, $A = \{a_1, a_2, ..., a_n\}$
▷ The agent who runs this algorithm is called *this agent*, A_{this}
▷ $PI^{A_{this}}$ is the profile information of A_{this}
▷ The participant agents of CHC is presented with $A^{participant} \subset A$

Result: A set of groups, $\{group_1, group_2, ..., group_k\}$. A group, $group_k = \{a_1, a_2, ..., a_t\}$ is a set of agents who are assigned to the same group.

1: **if** A_{this} not running a CHC process **then**
2: **Request** the profile information from all agents in $A/\{A_{this}\}$
3: **while** $true$ **do**
4: **if** $n - 1$ agents returned their profile information or 60 seconds passed **then**
5: Terminate (**while loop**)
6: **end if**
7: **end while**
8: **Broadcast** the profile information of all agents in ($A^{participant}$) to all agents in ($A^{participant}/\{A_{this}\}$)
9: **Compute** distance scores with equation 5.4
10: **while** $true$ **do**
11: **if** ($|A^{participant}| - 1$) agents returned their distance scores or 100 seconds passed **then**
12: Terminate (**while loop**)
13: **end if**
14: **end while**
15: **Compute** reciprocal scores with equation 5.6.
16: **Run** *HDBSCAN* algorithm to solve CHC and produce a grouping solution (i.e. set of groups)
17: **Save** the solution
18: **Broadcast** the solution to all agents in ($A^{participant}/\{A_{this}\}$)
19: **Send** the solution to the TCN of (A_{this})
20: **else**
21: **Inform** TCN that agent is already running a CHC process
22: **end if**

The runtime complexity of algorithm 3 is $T^{pi} + p * (n-1) + T^{ds} + n/2$ where T^{pi} and T^{ds} are the numbers of iterations to collect the profile information and distance scores from participant agents. Moreover, p and n are the numbers of CHC preferences and participant agents. $n/2$ is the complexity to compute reciprocal scores.

Algorithm 4 describes the steps of participant agents to return their profile information to the dedicated agent (line 2). Line 1 checks the consent of the TCN of the participant agent for joining a CHC process.

Algorithm 4 Participant agent's profile information

Input: The ID of the dedicated agent, $A_{dedicated}$
Result: Profile information, $PI^{A_{this}}$ of this agent

1: **if** TCN of A_{this} accepts joining a CHC process **then**
2: **Send** the profile information, $PI^{A_{this}}$ to $A_{dedicated}$
3: **end if**

Algorithm 5 is used by the participant agents to compute and return their distance scores when receiving a request from the dedicated agent.

Algorithm 5 Computation of participant agent's distance scores

Input: The ID of the dedicated agent, $A_{dedicated}$
Result: Distance scores set $\{(D^1, A_1), (D^2, A_2), ..., (D^n, A_n)\}$ where (D^n, A_n) represents the distance score, D^n between agents A_{this} and A_n

1: **Compute** distance scores with equation 5.4
2: **Send** distance scores to $A_{dedicated}$

CHC Coordination Assumptions

In the CHC use-case, the TCNs and agents had some limitations as listed below:

Assumption 1: All agents run without any technical (e.g. server-side) errors.

Assumption 2: No agent can join the coordination process after it was started.

Assumption 3: The modification of the configurations such as individual preferences, profile information, and importance weights of the preferences during a coordination process will be taken into account in the next coordination process.

Assumption 4: The coordination process cannot be stopped in the middle by the TCNs.

Assumption 5: If an agent does not return its profile information or distance scores, then its preferences will not be considered in the computed solutions. Not returning such data does not break the process.

Assumption 6: If the dedicated agent fails, the CHC process will fail. It will need to be restarted again

Assumption 7: Each TCN can run a single CHC process at a time

5.3 CHC Implementation

This section presents the implemented BTs for each step of the CHC coordination protocol without going into detail. Additionally, *AkbarKazimov/LCC-CHC-Use-cases-with-MAJAN* [22] GitHub repository provides the implementation of the CHC use-case.

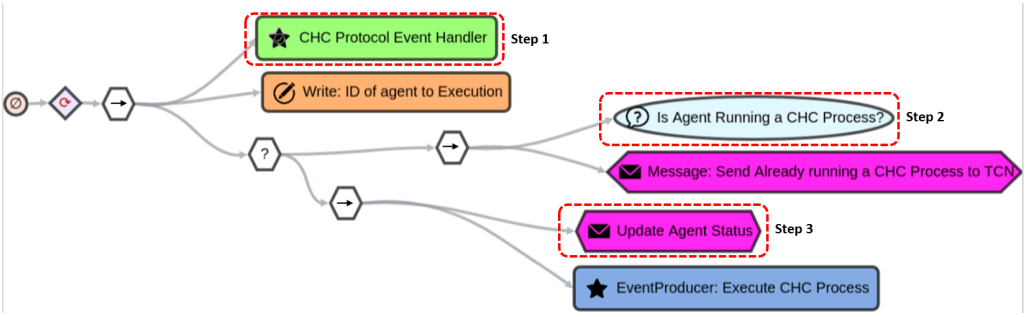

Figure 5.3: *Start CHC Protocol* BT covers steps 1, 2 and 3 of CHC protocol (figure 5.2)

Steps 1, 2, and 3 in figure 5.2 enabled the agents to receive the *start coordination* signal, check if the agent was running any CHC process, and update the agent status, respectively. Since these steps were specific to the CHC use-case, they were not provided in the clustering coordination protocol (figure 3.13). Therefore, they were implemented in the newly developed BT in figure 5.3.

[22] `https://github.com/AkbarKazimov/LCC-CHC-Use-cases-with-MAJAN/tree/main/CHC`

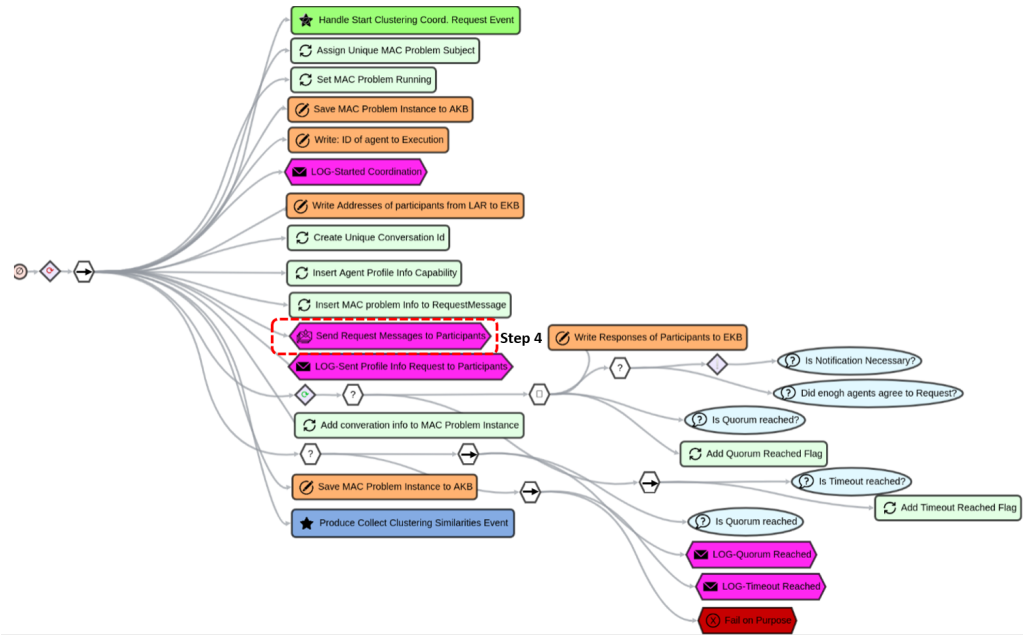

Figure 5.4: *Start Clustering Protocol* BT covers step 4 of CHC protocol (figure 5.2)

For the implementation of **step 4** which enabled the dedicated agent to request the profile information, the BT in figure 5.4 was used. It was provided by MAJAN as a template BT for the clustering coordination protocol.

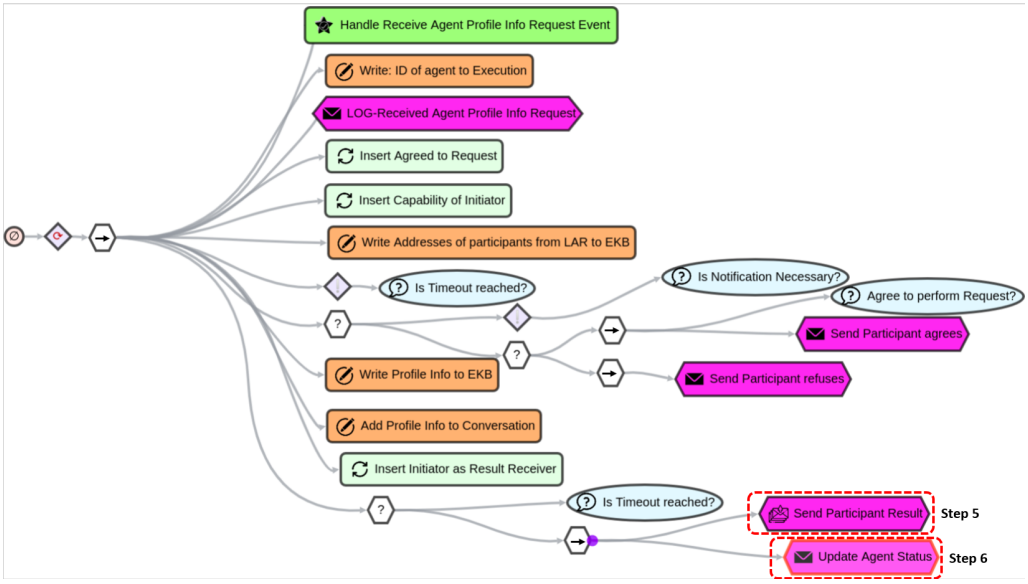

Figure 5.5: *Return Profile Information* BT covers steps 5 and 6 of CHC protocol (figure 5.2)

The BT in figure 5.5 implemented **steps 5** and **6** that were returning the profile information and updating the agent status, respectively. It was a clustering protocol BT. However, it was extended with the *Update Agent Status* message node since **step 6** was specific to the CHC use-case.

Step 7 broadcast the complete profile information to all participants, and it was implemented in the template clustering BT in figure 5.6.

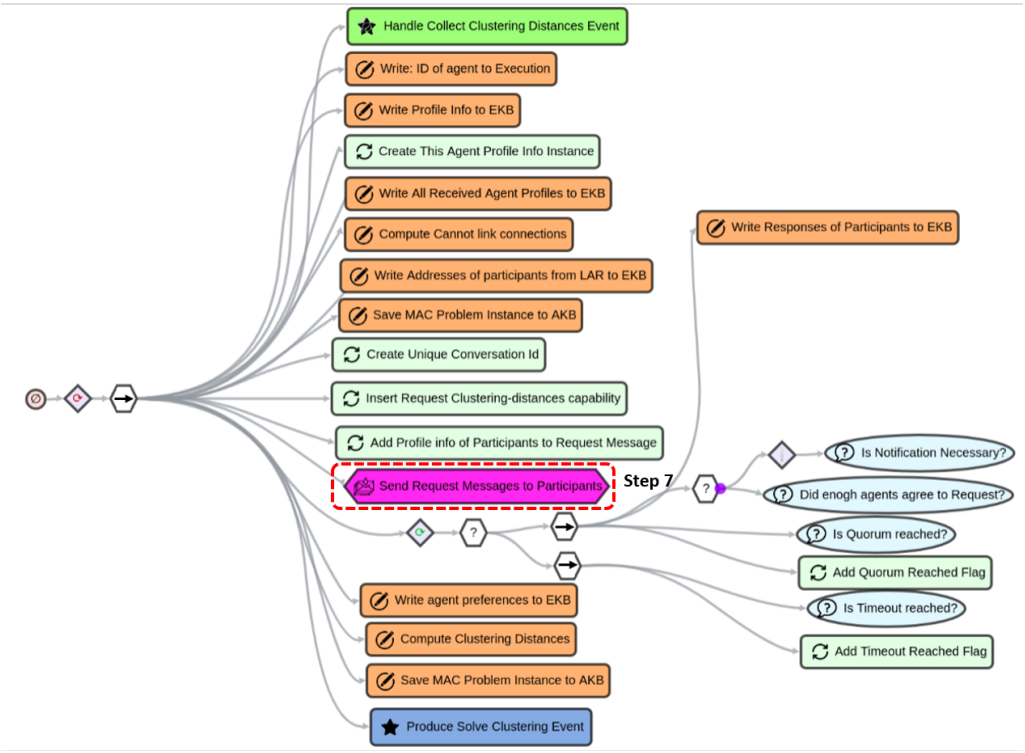

Figure 5.6: *Collect Clustering Distances* BT covers step 7 of CHC protocol (figure 5.2)

Figure 5.7: *Compute Clustering Distances* BT covers steps 8 and 9 of CHC protocol (figure 5.2)

The *Compute Clustering Distances* (**step 8**) and *Send Participant Result* (**step 9**) nodes were used to compute the distance scores and return them accordingly in the template BT in figure 5.7.

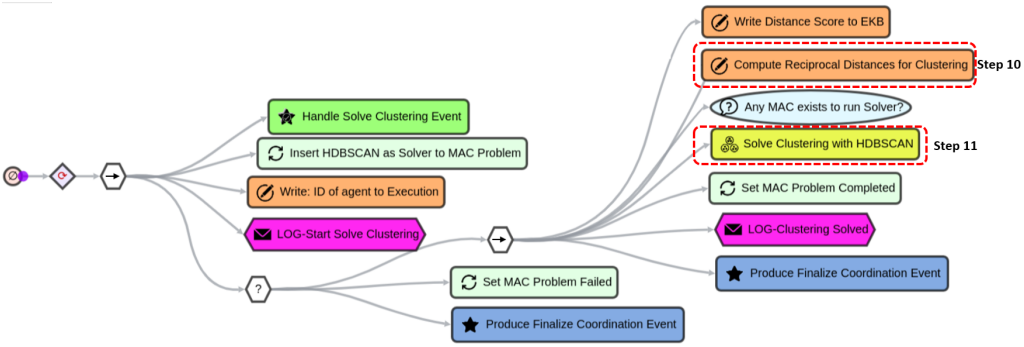

Figure 5.8: *Solve Clustering* BT covers the steps 10 and 11 of CHC protocol (figure 5.2)

The implementation of **steps 10** and **11** did not require any modification, since they were already covered in the template clustering protocol BT in figure 5.8.

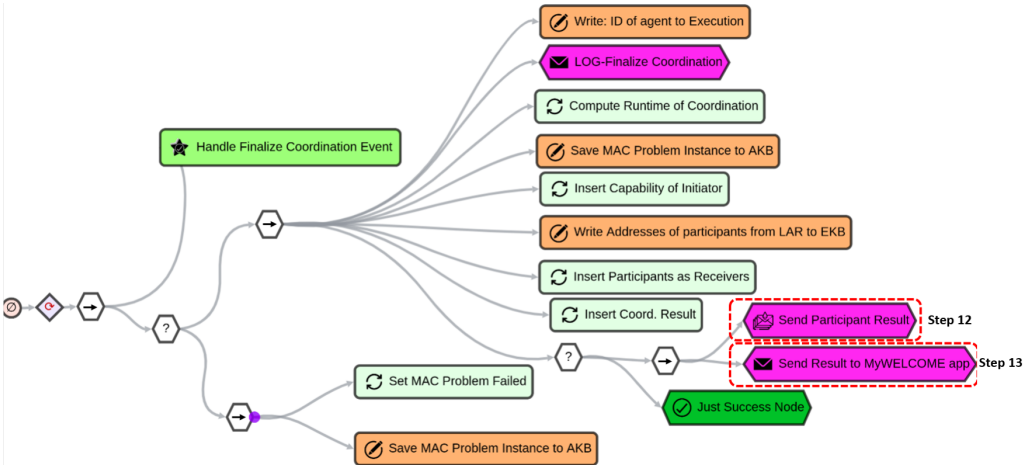

Figure 5.9: *Finalize Coordination* BT covers the steps 12 and 13 of CHC protocol (figure 5.2)

Step 12 broadcast the solution, and it was already implemented in the provided template BT in figure 5.9. This BT was extended with the *Send Result to MyWELCOME app* message node to implement **step 13** since it was a CHC-related step.

CHC feature in the MyWELCOME application

In the CHC use-case, only the MyWELCOME application was responsible for providing UI to the TCNs. By the submission date of this thesis, the CHC feature was not implemented in the MyWELCOME application, as the WELCOME project was ongoing.

5.4 CHC Evaluation

This section describes the evaluation of the CHC use-case. The process of matching the TCNs based on their preferences for finding an apartment to share had been performed manually so far by the user partner PRAKSIS [23] who provided the CHC use-case. Therefore, the same evaluation approach was applied (i.e. the Customer Satisfaction Score (CSAT)) to measure the satisfaction of the users with the provided features on a Likert scale (table 4.1).

Moreover, additional TCN profiles were created since the LCC and CHC use-cases have different individual preferences and profile information requirements. The next section introduces the settings and the results of the evaluation process.

5.4.1 Settings

Five data records consisting of 15, 20, 25, 50, and 100 TCN profiles were created as the settings of this evaluation. Each profile had the following attributes:

- **Demographics**: *age*, *gender*, *family*, *nationality*, *religion*, and *ethnicity*.

[23] `https://praksis.gr/`

- **Individual Preferences**: *age preference*, *gender preference*, *family preference*, *nationality preference*, *religion preference*, *ethnicity preference*, *location preference*, *accessibility preference*, *rent period preference*, and *share-with preference*.

Tables 5.2 and 5.3 describe the demographics and the preferences of the TCNs for data record 1, respectively.

ID	Age	Gender	Family	Nationality	Religion	Ethnicity
1	39	Female	Extended	Nat7	Rel4	Eth3
2	37	Male	Single Man	Nat1	Rel1	Eth2
3	39	Male	Single Parent Father	Nat7	Rel2	Eth3
4	52	Other	Single Parent Father	Nat2	Rel7	Eth1
5	70	Other	Single Woman	Nat5	Rel4	Eth2
6	71	Other	Extended	Nat1	Rel2	Eth3
7	24	Male	Single Man	Nat3	Rel6	Eth2
8	47	Other	Extended	Nat4	Rel3	Eth3
9	54	Male	Single Man	Nat4	Rel3	Eth6
10	51	Female	Single Woman	Nat5	Rel7	Eth7
11	71	Other	Nuclear	Nat7	Rel7	Eth7
12	62	Male	Single Man	Nat1	Rel4	Eth5
13	30	Other	Nuclear	Nat1	Rel3	Eth2
14	63	Male	Single Man	Nat2	Rel6	Eth4
15	56	Female	Nuclear	Nat2	Rel5	Eth5

Table 5.2: CHC data record 1: Demographic data of TCNs

ID	Age P.	Gender P.	Family P.	Nationality P.	Religion P.	Ethnicity P.	Location P.	Accessibility P.	Rent Period P.	Share-with P.
1	44-50	Female	Single Woman/Single Mother	Mixed	Don't mind	Mixed	L2/L4/L7	No	1/3/2021-1/7/2022	2-3
2	34-43	Don't mind	Nuclear	Don't mind	Don't mind	Don't mind	L1	Yes	1/1/2021-1/7/2022	5
3	44-50	Female	Single Woman/Single Mother	Mixed	Don't mind	Mixed	L3	Don't mind	1/1/2021-1/7/2022	5
4	44-50	Other	Don't mind	Same	Mixed	Same	L4	No	1/1/2021-1/4/2021	2-3
5	34-43/65-120	Female	Extended/Nuclear	Don't mind	Don't mind	Mixed	L2/L3	Yes	3/7/2021-12/2/2022	5
6	65-120	Other	Single Woman/Single Mother	Mixed	Mixed	Same	L3/L6	Yes	1/1/2021-1/7/2021	2-7
7	18-25/26-33/44-50	Male	Extended/Nuclear	Mixed	Don't mind	Don't mind	L4	Don't mind	1/1/2021-1/7/2022	2-7
8	44-50	Don't mind	Single Mother	Same	Mixed	Mixed	L4/L2/L7	Don't mind	1/1/2021-1/7/2022	5
9	Don't mind	Don't mind	Extended	Mixed	Same	Same	L4/L2/L7	No	1/5/2021-1/5/2022	3-5
10	44-50/51-65	Male	Single Man	Don't mind	Same	Mixed	L5	Yes	15/3/2021-1/4/2022	3-5
11	18-25/26-33	Female	Single Woman/Single Mother	Same	Don't mind	Mixed	L1/L2/L3	No	15/3/2021-1/4/2022	1-4
12	51-65	Female	Single Woman/Single Mother	Mixed	Mixed	Don't mind	L4	Yes	1/5/2021-1/5/2022	2-4
13	34-43	Other	Don't mind	Same	Don't mind	Don't mind	L4/L2/L7	Yes	1/5/2021-1/5/2022	2-3
14	18-25/26-33	Male	Single Mother/Single Father/Extended	Don't mind	Don't mind	Don't mind	L2/L3	Don't mind	23/4/2020-17/5/2021	2
15	Don't mind	Female	Single Mother	Don't mind	Don't mind	Same	L3/L6	Don't mind	1/9/2021-1/12/2021	3-5

Table 5.3: CHC data record 1: Individual Preferences of TCNs

Furthermore, PRAKSIS provided three configurations that consisted of different importance weights of the individual preferences:

1. The first configuration is described in table 5.1.
2. In the second configuration in table 5.4, *age*, *gender*, and *nationality* preferences were very important compared to the apartment preferences.

Individual Preference	Importance Weight
1. Age	10
2. Gender	10
3. Family	3
4. Nationality	10
5. Religion	8
6. Ethnicity	9
7. Location of Apartment	1
8. Accessibility for Disabled	1
9. Rental Period	1
10. Share with	2

Table 5.4: Second configuration for CHC

3. In the third configuration in table 5.5, the apartment preferences were more important compared to configurations 1 (table 5.1) and 2 (table 5.4).

Individual Preference	Importance Weight
1. Age	4
2. Gender	7
3. Family	5
4. Nationality	7
5. Religion	7
6. Ethnicity	9
7. Location of Apartment	5
8. Accessibility for Disabled	4
9. Rental Period	7
10. Share with	4

Table 5.5: The third configuration for CHC

5.4.2 Results

To identify the final evaluation score, 15 results were computed by agents through the execution of each of 5 data records with each of 3 configurations. The results, data records, and configurations were provided to PRAKSIS to collect their CSAT scores for each result. The final evaluation score, **4.3**, was calculated as the arithmetic mean of the CSAT scores of all the results, and it corresponded to the *satisfied* on the Likert scale (table 4.1). This section presents only the results of the first data record that was executed with all 3 configurations. The rest of the results are provided in appendix D.

Results of Data Record 1 (tables 5.2 and 5.3): A solution is a grouping that consists of groups and singletons (i.e. outliers) in the following results. The singletons are the TCNs who are unassigned to any group. The numbers in the solutions represent the agent/TCN IDs.

1. **Configuration 1 (table 5.1)**
   - **Total Computation time**: 63 seconds
   - **Solution**:
     Singletons: [1, 4, 8, 11]
     Group 1: [7,14]

Group 2: [9, 10, 12, 15]
Group 3: [2, 13]
Group 4: [3, 5, 6]

- **Customer Satisfaction Score**: 4

2. **Configuration 2 (table 5.4)**
   - **Total Computation time**: 71 seconds
   - **Solution**:
     Singletons: [9]
     Group 1: [4, 8, 11]
     Group 2: [3, 5, 6]
     Group 3: [1, 10, 12, 15]
     Group 4: [2, 13]
     Group 5: [7, 14]
   - **Customer Satisfaction Score**: 4.17
3. **Configuration 3 (table 5.5)**
   - **Total Computation time**: 62 seconds
   - **Solution**:
     Singletons: [13]
     Group 1: [1, 4, 9, 11, 15]
     Group 2: [10, 12]
     Group 3: [3, 5, 6]
     Group 4: [2, 7, 8, 14]
   - **Customer Satisfaction Score**: 4.2

Chapter 6
Conclusion

This thesis introduced the problem of distributed coordination of SPARQL-Behavior-Tree-based (SPARQL-BT) AJAN agents into groups at the beginning. Throughout this work, a methodological software development approach was executed which produced a functioning software product by extending the existing AJAN with MAJAN that provides new features for the realization and evaluation of Multiagent Coordination (MAC) use-cases. This chapter summarizes the results of this work and takes a look at the goals from section 1.2. Moreover, the limitations of this work and the potential future work are also described in the subsections.

6.1 Summary

Multiagent systems provide lots of benefits such as autonomy, scalability, concurrency, adaptability, and so on, and they have been used in various applications as stated in section 1.1. Two of the important problems in Multiagent Systems (MAS) are the coordination of actions of the agents and the formation of groups of agents. To design MASs, AJAN (section 2.1) has been developed at DFKI.

AJAN agents are SPARQL-BT-based, meaning that the behaviors of AJAN agents are modeled with BTs and these BTs use SPARQL queries to manipulate the agent knowledge, since it is represented as a Resource Description Framework (RDF). AJAN does not provide features as the solutions to these problems to coordinate agents into groups. This thesis aims to solve this problem by extending AJAN to provide features for AJAN users to design MAC use-cases.

To produce a working software product, this thesis applied the user-centered Software Development Life Cycle (SDLC) model so that the requirements of the potential users could be collected and delivered as well as possible. According to the requirements, users expected template generic SPARQL-BTs so that they can design the coordination behaviors of agents easily for the realization of MAC use-cases. Moreover, they expected features for the visualization and evaluation of the grouping results of the MAC use-cases.

One iteration of the *Requirements - Design - Implementation - Evaluation* cycle was executed in this work. Firstly, the requirements were collected via interviews with users

and from the goals of this thesis. Secondly, the concepts and UI mockups were designed based on these requirements. Then, the concepts and the mockups were implemented to provide functional software, which was then evaluated in the last phase by conducting surveys with potential users. According to the answers of the users in the evaluation surveys, the System Usability Scale (SUS) [22, 45] score was computed as 83.5 which corresponds to *Excellent* in table 3.1.

Additionally, the feedback that was collected in the evaluation surveys is described in section 3.2.5. Although one iteration of the SDLC model was executed in this thesis, 9 of 11 requirements that were collected during the evaluation phase were implemented as well. Section 3.2.6 presents the final version of MAJAN. To summarize, MAJAN provides the following features:

- Template generic coordination SPARQL-BTs and coordination nodes to implement the FIPA Request, Coalition Structure Generation Problem (CSGP), and clustering coordination protocols for the realization of MAC use-cases.
- MAJAN web for the visualization and comparison of the execution results of MAC processes and centrally running grouping algorithms.
- Postman collections to create agents and execute coordination protocols in an efficient and faster way.

Now, MAJAN provides all the features which the other agent engineering tools are missing, as summarized in section 2.3.3. Moreover, this work provides a **MAJAN Guideline** document that gives the instructions and explanations of all the functionalities of MAJAN. The source code of MAJAN, template SPARQL-BTs, Postman collections, guideline document, and more are provided in the public MAJAN GitHub repository [24].

Besides, Language Course Coordination (LCC) (section 4) and Co-Habitation Coordination (CHC) (section 5) coordination use-cases of the MyWELCOME application were implemented and evaluated as part of this work. For their implementation, the template coordination BTs that MAJAN provides were used. The use-cases were evaluated by user partners DIFE [25] and PRAKSIS [26] who were the providers of the LCC and CHC, respectively. According to the evaluation results, the DIFE users were *very satisfied* (5 out of 5) with the LCC results (section 4.4), and the PRAKSIS users were *satisfied* (4.3 out of 5) with the CHC results (section 5.4).

As stated in section 1.2, this thesis had two goals. Successful implementation and evaluation of MAJAN and the given coordination use-cases show that both of the goals are achieved.

6.2 Limitations and Future Work

Because of the limited time of this thesis work, the SDLC model was executed once. Each phase of the development model requires lots of time and work. Additionally, the second goal (i.e. implementation and evaluation of the given use-cases) of this work was as important as the first goal (i.e. implementation of MAJAN). Because these use-cases were very important parts of the WELCOME project, they had to be delivered, which required a lot of time.

Another limitation was the execution technique of **Goals** in AJAN. In AJAN, **Goals** are similar to **Events** such that both of them are attached to specific BTs so that whenever a

[24] `https://github.com/AkbarKazimov/MAJAN`
[25] `https://web.gencat.cat/ca/inici`
[26] `https://praksis.gr/`

Goal or an Event is called, it triggers the respective BT. The difference between the two is that Events work asynchronously, while Goals are synchronous. Meaning that when a BT calls a Goal, it will continue to its execution only after the BT that is attached to the called Goal finishes. Meanwhile, in the case of Events, the BT that calls the Event proceeds with its execution without waiting for anything. Therefore, the Goals could be used to trigger the template coordination BTs that MAJAN provided so that the process of modeling coordination behaviors of agents could be even more user-friendly. However, the limitation was that a single Goal and the BT that is attached to it cannot be executed multiple times in parallel. This parallelism is crucial for the MAC because agents must be able to participate in multiple coordination processes at the same time. The Events in AJAN, on the other hand, support this parallelism. In future work, the Goals can be modified to support the parallelism so that the template coordination or any other BTs that are required to be executed in parallel can be provided as Goals.

MAJAN provides Postman collections to easily create multiple agents and send messages in parallel by specifying dynamic information in configuration files. The necessary instructions to run them are provided in the MAJAN Guideline document. In future work, AJAN Editor can be extended appropriately to support the functionalities of creating multiple agents in parallel by giving the agent ID and their initial knowledge dynamically and sending the same request to multiple agents with a single click.

Bibliography

[1] Abhinav, A. (2010). Postman api platform. *https://www.postman.com/*. Accessed: 06-07-2022.

[2] Agis, R. A., Gottifredi, S., and García, A. J. (2020). An event-driven behavior trees extension to facilitate non-player multi-agent coordination in video games. *Expert Systems with Applications*, 155:113457. Publisher: Elsevier.

[3] Ahmed, O. (2014a). Ember-cli-chart. *https://github.com/aomran/ember-cli-chart*. Accessed: 06-07-2022.

[4] Ahmed, S. (2014b). Ember-highcharts. *https://github.com/ahmadsoe/ember-highcharts*. Accessed: 06-07-2022.

[5] Aknine, S., Ramchurn, S., Changder, N., and Dutta, A. (2021). Boss: A bi-directional search technique for optimal coalition structure generation with minimal overlapping, student abstract. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[6] Alfonso, B., Vivancos, E., Botti, V., and García-Fornes, A. (2011). Integrating jason in a multi-agent platform with support for interaction protocols. In *Proceedings of the compilation of the co-located workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, & VMIL'11*, pages 221–226.

[7] Alibhai, Z. (2003). What is contract net interaction protocol. *Isotope Ratio Mass Spectrometry (IRMS) Lab*, page 18.

[8] Alishevskikh, A. (2017). Rdfbeans. *https://rdfbeans.github.io/quickstart.html*. Accessed: 06-07-2022.

[9] Alqurashi, T. and Wang, W. (2019). Clustering ensemble method. *Machine Learning and Cybernetics*, 10(6):1227–1246. Publisher: Springer.

[10] Antakli, A., Zinnikus, I., Ebersold, J., Herget, W., and Spieldenner, D. (2019). Ajan: Accessible java agent nucleus. *https://asr.dfki.de/ajan/*, *https://github.com/aantakli/AJAN-service*, *https://github.com/aantakli/AJAN-editor*. Accessed: 06-07-2022.

[11] Any, C. (2016). Ember-anychart. *https://github.com/AnyChart/AnyChart-Ember*. Accessed: 06-07-2022.

[12] Ayeni, A. (2014). Empirics of standard deviation. In *Conference: Research Presentation at Covenant University*.

[13] Beckett, D. (2014a). Rdf 1.1 n-triples. *https://www.w3.org/TR/2014/REC-n-triples-20140225/*. Accessed: 06-07-2022.

[14] Beckett, D. (2014b). Rdf 1.1 turtle. *https://www.w3.org/TR/2014/REC-turtle-20140225/*. Accessed: 06-07-2022.

[15] Beckett, D. (2014c). Rdf 1.1 xml syntax. *https://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/*. Accessed: 06-07-2022.

[16] Bergenti, F., Poggi, A., Rimassa, G., Turci, P., and Tomaiuolo, M. (2005). Developing agent-based applications with jade. In *Design of Intelligent Multi-Agent Systems*, pages 191–214. Publisher: Springer.

[17] Bizer, C. (2014). Rdf 1.1 trig. `https://www.w3.org/TR/2014/REC-trig-20140225/`. Accessed: 06-07-2022.

[18] Bojic, I., Lipic, T., Kusek, M., and Jezic, G. (2011). Extending the jade agent behaviour model with jbehaviourtrees framework. In *International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, pages 159–168. Publisher: Springer.

[19] Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming multi-agent systems in AgentSpeak using Jason*. Publisher: John Wiley & Sons.

[20] Braubach, L., Pokahr, A., and Lamersdorf, W. (2005). Jadex: A bdi-agent system combining middleware and reasoning. In *Software agent-based applications, platforms and development kits*, pages 143–168. Publisher: Springer.

[21] Brendan, E. (1995). Javascript. `https://developer.mozilla.org/en-US/docs/Web/JavaScript`. Accessed: 06-07-2022.

[22] Brooke, J. (1996). Sus: a "quick and dirty'usability. *Usability evaluation in industry*, 189(3). Publisher: London–.

[23] Campello, R. J., Moulavi, D., Zimek, A., and Sander, J. (2015). Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(1):1–51. Publisher: ACM New York, NY, USA.

[24] Carothers, G. (2014). Rdf 1.1 n-quads. `https://www.w3.org/TR/2014/REC-n-quads-20140225/`. Accessed: 06-07-2022.

[25] Castelfranchi, C. (1995). Guarantees for autonomy in cognitive agent architecture. In *Intelligent Agents*. Publisher: Springer Berlin Heidelberg.

[26] Changder, N., Aknine, S., Ramchurn, S., and Dutta, A. (2020). Odss: Efficient hybridization for optimal coalition structure generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7079–7086.

[27] Colledanchise, M. and Ögren, P. (2018). *Behavior trees in robotics and AI: An introduction*. Publisher: Chemical Rubber Company (CRC) Press.

[28] Collier, R. W., Russell, S., and Lillis, D. (2015). Reflecting on agent programming with agentspeak (l). In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 351–366. Publisher: Springer.

[29] Consortium, W. W. W. (1996). Css. `https://developer.mozilla.org/en-US/docs/Web/CSS`. Accessed: 06-07-2022.

[30] Dahl, R. (2009). Node. js. In `https://s3.amazonaws.com/four.livejournal/20091117/jsconf.pdf`. Accessed: 06-07-2022.

[31] Dinh, H. T., Lee, C., Niyato, D., and Wang, P. (2013). A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless communications and mobile computing*. Publisher: Wiley Online Library.

[32] EES (2018). Mobile cloud computing: What, how, and why? `https://www.eescorporation.com/mobile-cloud-computing/`. Accessed: 06-07-2022.

[33] Ertl, B., Meyer, J., Schneider, M., and Streit, A. (2020). Coexdbscan: Density-based clustering with constrained expansion. In *Knowledge Discovery and Information Retrieval (KDIR)*, pages 104–115.

[34] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Knowledge Discovery and Data Mining (KDD)*, volume 96, pages 226–231.

[35] Estévez, P. A., Tesmer, M., Perez, C. A., and Zurada, J. M. (2009). Normalized mutual information feature selection. *Transactions on neural networks*, 20(2):189–201. Publisher: Institute of Electrical and Electronics Engineers (IEEE).

[36] FIPA, T. (2002a). Communication, fipa english auction interaction protocol specification, foundation for intelligent physical agents. `http://www.fipa.org/specs/fipa00031/index.html`. Accessed: 06-07-2022.

[37] FIPA, T. (2002b). Communication, fipa request interaction protocol specification, foundation for intelligent physical agents. `http://www.fipa.org/specs/fipa00026`. Accessed: 06-07-2022.

[38] Froala (2017). Ember-froalacharts. `https://github.com/froala/ember-froalacharts`. Accessed: 06-07-2022.

[39] Gearon, P. (2013). Sparql 1.1 update. `https://www.w3.org/TR/2013/REC-sparql11-update-20130321/`. Accessed: 06-07-2022.

[40] Gonsai, A. and Raval, M. R. (2013). Mobile cloud computing: A tool for future. *Computer Science & Engineering Technology (IJCSET), ISSN*. Publisher: Citeseer.

[41] Harris, S. (2013). Sparql 1.1 query language. `https://www.w3.org/TR/sparql11-query/describe`. Accessed: 06-07-2022.

[42] Hayes, P. (2014). Blank nodes. `https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/\protect/\penalty\exhyphenpenaltyblank-nodes`. Accessed: 06-07-2022.

[43] Ivan, V. (2015). Ember-d3. `https://www.npmjs.com/package/ember-d3`. Accessed: 06-07-2022.

[44] JavaScript (2021). Amcharts5: Javascript charts and maps. `https://www.amcharts.com/`. Accessed: 06-07-2022.

[45] Jeff, S. (2018). 5 ways to interpret a sus score. `https://measuringu.com/interpret-sus-score/`. Accessed: 06-07-2022.

[46] Kazemi, H. M., Liasi, S. G., and Sheikh-El-Eslami, M. (2018). Generation expansion planning considering investment dynamic of market participants using multi-agent system. In *2018 Smart Grid Conference (SGC)*, pages 1–6. Publisher: Institute of Electrical and Electronics Engineers (IEEE).

[47] Kazimov, A. (2022). Majan: Multiagent coordination with ajan. `https://github.com/AkbarKazimov/MAJAN`. Accessed: 06-07-2022.

[48] Kellogg, G. (2014). Json-ld 1.1. `https://www.w3.org/TR/json-ld/`. Accessed: 06-07-2022.

[49] Lazar, J. and Barbosa, S. D. (2017). Introduction to human-computer interaction. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. Publisher: Association for Computing Machinery.

[50] MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Publisher: Oakland, CA, USA.

[51] Malone, T. W. and Crowston, K. (1994). The interdisciplinary study of coordination. *Association for Computing Machinery (ACM) Comput. Surv.* Publisher: Acm New York, NY, USA.

[52] Mariani, S., Omicini, A., and Sangiorgi, L. (2015). Models of autonomy and coordination: integrating subjective and objective approaches in agent development frameworks. In *Intelligent Distributed Computing VIII*, pages 69–79. Publisher: Springer.

[53] Mittal, V. and Frennea, C. (2010). Customer satisfaction: a strategic review and guidelines for managers. *Marketing Science Institute (MSI) Fast Forward Series, Cambridge, MA*.

[54] Nielsen, F. (2016). Hierarchical clustering. In *Introduction to High Performance Computing (HPC) with Message Passing Interface (MPI) for Data Science*, pages 195–211. Publisher: Springer.

[55] Nwana, H. S., Lee, L., and Jennings, N. R. (1997). Co-ordination in multi-agent systems. *Software agents and soft computing towards enhancing machine intelligence*, pages 42–58. Publisher: Springer.

[56] Okabe, M. and Yamada, S. (2018). Clustering using boosted constrained k-means algorithm. *Frontiers in Robotics and AI*, 5:18. Publisher: Frontiers.

[57] Oprea, M. (2004). Applications of multi-agent systems. In *Information technology*, pages 239–270. Publisher: Springer.

[58] Palomares, I., Porcel, C., Pizzato, L., Guy, I., and Herrera-Viedma, E. (2021). Reciprocal recommender systems: Analysis of state-of-art literature, challenges and opportunities towards social recommendation. *Information Fusion*, 69:103–127. Publisher: Elsevier.

[59] PF4J, P. F. (2012). Pf4j–plugin framework for java. `https://pf4j.org/`. Accessed: 06-07-2022.

[60] Pitt, J., Kamara, L., Sergot, M., and Artikis, A. (2006). Voting in multi-agent systems. *The Computer Journal*, 49(2):156–170. Publisher: Oxford University Press.

[61] Pizzato, L., Rej, T., Chung, T., Koprinska, I., and Kay, J. (2010). Recon: a reciprocal recommender for online dating. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 207–214.

[62] Qi, H. and Gani, A. (2012). Research on mobile cloud computing: Review, trend and perspectives. In *International Conference on Digital Information and Communication Technology and it's applications (DICTAP)*. Publisher: Institute of Electrical and Electronics Engineers (IEEE).

[63] Rahwan, T., Michalak, T. P., Wooldridge, M., and Jennings, N. R. (2015). Coalition structure generation: A survey. *Artificial Intelligence*, 229:139–174. Publisher: Elsevier.

[64] RDF4J, E. (2002). Eclipse rdf4j-a java framework for rdf. `http://rdf4j.org`. Accessed: 06-07-2022.

[65] Ruparelia, N. B. (2010). Software development lifecycle models. *ACM Special Interest Group on Software Engineering (SIGSOFT) Software Engineering Notes*, 35(3):8–13. Publisher: ACM New York, NY, USA.

[66] Singh, V. P., Kishor, N., and Samuel, P. (2017). Distributed multi-agent system-based load frequency control for multi-area power system in smart grid. *Transactions on Industrial Electronics*, 64(6):5151–5160. Publisher: Institute of Electrical and Electronics Engineers (IEEE).

[67] Srour, J., Tamas, M., De Weerdt, M., and Zuidwijk, R. (2008). Can agents measure up? a comparative study of an agent-based and on-line optimization approach for a drayage problem with uncertainty. *Transportation Research Part C: Emerging Technologies*. Publisher: Elsevier.

[68] Suguri, H. (1999). A standardization effort for agent technologies: The foundation for intelligent physical agents and its activities. In *Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences. 1999. HICSS-32. Abstracts and CD-ROM of Full Papers*, pages 10–pp. Publisher: Institute of Electrical and Electronics Engineers (IEEE).

[69] Sycara, K. (1998). Multiagent systems. *AI Magazine*, 19:79–92.

[70] Taguelmimt, R., Aknine, S., Boukredera, D., and Changder, N. (2021). Code-based algorithm for coalition structure generation. In *33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1075–1082. Publisher: Institute of Electrical and Electronics Engineers (IEEE).

[71] Tim, B.-L. (1993). Html. `https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics`. Accessed: 06-07-2022.

[72] Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S., et al. (2001). Constrained k-means clustering with background knowledge. In *International Conference on Machine Learning (ICML)*, volume 1, pages 577–584.

[73] Wanner, L. (2020). Reception, management and integration of third party nationals. `https://welcome-h2020.eu/assets/docs/factsheet_welcome_2020_english.pdf`. Accessed: 06-07-2022.

[74] Warwas, S., Fischer, K., Klusch, M., and Slusallek, P. (2012). Bochica: A model-driven framework for engineering multiagent systems. In *International Conference on Agents and Artificial Intelligence (ICAART) (1)*, pages 109–118.

[75] Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C., and Deleuze, S. (2013). Spring boot reference guide. *Part IV. Spring Boot features*, 24.

[76] Weiss, G. (1999). A modern approach to distributed artificial intelligence. *Institute of Electrical and Electronics Engineers (IEEE) Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 22(2).

[77] Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: theory and practice. In *The Knowledge Engineering Review*. Publisher: Cambridge University Press.

[78] Woolridge, M. and Wooldridge J., M. (2001). *Introduction to multiagent systems*. Publisher: John Wiley & Sons, Inc.

[79] Xie, X.-F., Smith, S. F., and Barlow, G. J. (2012). Schedule-driven coordination for real-time traffic network control. In *Twenty-Second International Conference on Automated Planning and Scheduling*.

[80] Yacef, K. and McLaren, B. M. (2015). Supporting learners' group formation with reciprocal recommender technology. In *3rd Workshop on Intelligent Support for Learning in Groups, held in conjunction with ITS*.

[81] Yehuda, K. (2011). Ember js. `https://emberjs.com/`. Accessed: 06-07-2022.

Appendix A
MAJAN User Requirements Survey

This appendix provides the questions that were asked in the *Requirements Elicitation Survey* of MAJAN. This survey was divided into **User Context**, **Multiagent Coordination Use-Case Realization**, **Multiagent Coordination Use-Case Monitoring and Evaluation**, and **MAJAN Guideline** sections where each of which contained concepts and related questions. Google Forms [27] was used to prepare the questionnaire.

A.1 User Context Questions

This section asked the following questions to collect the background knowledge of users.

1. What is the topic of your study/profession?
2. What are your areas of expertise at DFKI?
3. Are you familiar with Intelligent Agents? If so, in which context?
4. Which development frameworks have you used/know to develop agents?
5. How familiar are you with AJAN?
6. Have you ever created an agent with AJAN? If so, in which context?
7. How familiar are you with RDF?
8. How familiar are you with SPARQL-BTs?
9. How experienced are you with JAVA?
10. Have you ever used AJAN Editor? If so, in which context?
11. How experienced are you with JavaScript?
12. Have you ever extended AJAN Editor? If so, what was your extension?

[27] `https://docs.google.com/forms/u/0/?tgif=d`

13. Have you ever created a new SPARQL-BT node in AJAN Editor? If so, in which context have you used it?
14. Have you ever used AJAN Plugin system? If so, in which context have you used it?
15. Are you familiar with the concept of Multiagent Coordination (MAC)? if so, in which context?
16. Are you familiar with Coordination Models? If so, which coordination models can you specify?
17. Are you familiar with interaction protocols? if so, which interaction protocols can you specify?
18. Have you ever used any Interaction Protocol? If so, in which context?
19. Are you familiar with Agent Communication Languages (ACL)? If so, which ACL can you specify? In which context have you used it?
20. Are you familiar with Coalition Structure Generation Problem (CSGP)? If so, which algorithms can you specify to solve a CSGP?
21. Are you familiar with Constraint Clustering problem? If so, which algorithms can you specify to solve a Constrained Clustering?
22. Are you familiar with any other coordination problem/algorithm? If so, which?

A.2 Multiagent Coordination Use-Case Realization Questions

This section presents the description of concepts that were provided in the survey to realize a MAC use-case and the questions to collect participants' feedback.

Concept 1: Generic SPARQL BT Nodes

To solve a problem, there will be specific predefined BT Node(s) in AJAN Editor provided as a feature of MAJAN. For example, to solve a CSGP, there will be 2 predefined nodes: 1) Generate Coalition Structures (CS) and 2) Run CSGP Solver algorithm. Each node will expect the input to be given via SPARQL queries. The nodes will be use-case independent, such that they can be used in a generic way regardless of the use-case at hand.

1. If this approach (having generic BT nodes) is not sufficient to solve CSGP (for LCC) and Constrained Clustering (for CHC) problems in a generic way, then how would you like it to be modified?
2. Would you like to add anything more?

Concept 2: Algorithm Integration

A problem can be solved with different algorithms. For example, CSGP can be solved with BOSS, FACS, ODP-IP, etc. In order to be able to make use of an algorithm to solve a problem, the algorithm will be integrated into AJAN via Plugin System. Afterward, the algorithm will be connected to the generic node which will solve the problem. In order not to restrict each generic node to a single algorithm, the name of the algorithm will be expected to be given as input via the SPARQL query.

1. Is this approach (described above) enough to be able to solve a problem with different algorithms which are specified in the agent knowledge? If no, how would you like it to be modified?
2. Is it a good approach to integrate the source code of the solver algorithms via Plugin System? If no, how would you like it to be modified?
3. Would you like to add anything more?

Concept 3: Agent-to-Agent Interaction

For the agents to be able to coordinate with each other, they need to have appropriate behaviors (BTs). Such behaviors should be designed by users, and they should reflect the MAC protocol of the use-case at hand. In other words, coordination protocol describes the steps that agents should follow to solve the use-case. Although agents do not have to communicate to coordinate, communication is one of the most important aspects of agent coordination.

1. Would it be useful for you to have any predefined interaction protocol in AJAN Editor? If so, which interaction protocols and in which context?

   Interaction Protocols (e.g. Contract Net, Auction, etc.) are typical patterns of message exchange between agents (i.e. structured conversations). In the current state of the AJAN, no predefined interaction protocol is provided.
2. Is the current communication architecture in AJAN (explained below) enough for multiagent communication from your point of view? if not, how would you like it to be modified in terms of MAC?

   In AJAN, the communication can be done via Action and Message nodes. None of them comply with any Agent Communication Languages (neither FIPA ACL nor KQML). This provides a flexible communication architecture, whether it is between two agents or an agent and an external software component, such that an agent designer is free to specify his/her capabilities and contents. In other words, agent designers do not have to use predefined performatives for agent-to-agent communication unlike in FIPA ACL [28] *and KQML* [29]
3. Would you like to add anything more?

Concept 4: Template BTs for MAC Implementation

As described in figure A.1, to design BTs in AJAN Editor, users need to select an icon from the section on the left (Graph Manipulation), drop it in the middle area, and connect the new icon with the existing ones with arrows.

[28] http://www.fipa.org/repository/aclspecs.html

[29] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.30.9451&rep=rep1&type=pdf

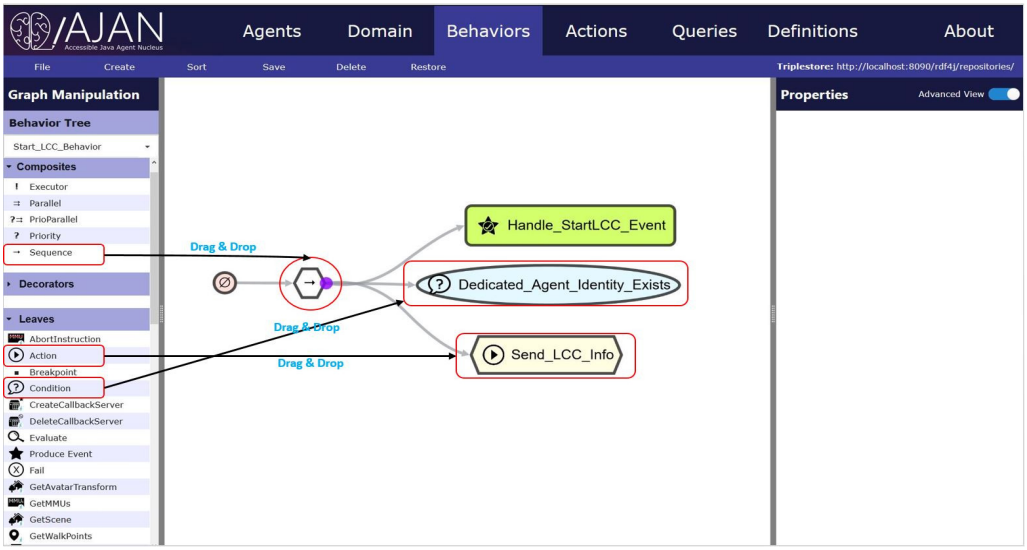

Figure A.1: Modelling BTs in AJAN Editor

LCC and CHC use-cases of the Welcome project will be solved as CSG and Constrained Clustering problems, respectively. The generic solutions for these two types of problems will be provided as templates as a feature of MAJAN. In other words, MAJAN will provide template BTs to solve these two problems in a use-case independent way. The template BTs will be available in AJAN Editor for users to select and customize depending on the needs of the use-case at hand.

Figure A.2 describes the way that AJAN Editor displays the Template BTs. The BTs which are listed in the area shown as #1 can be selected and customized by users. The area, shown as #2, lists the BTs which can be used as leaf nodes in BTs by Drag & Dropping. It is necessary to note that these BTs are not customizable when they are being used as leaf nodes. For example, Generate_CS_Behavior and Solve_CSGP_Behavior will be generic BTs to solve CSGP problems, and they can be modified by users depending on the use-case they have. For example, users can write SPARQL queries to specify the hard constraints, compute Utility values of the use-case, etc.

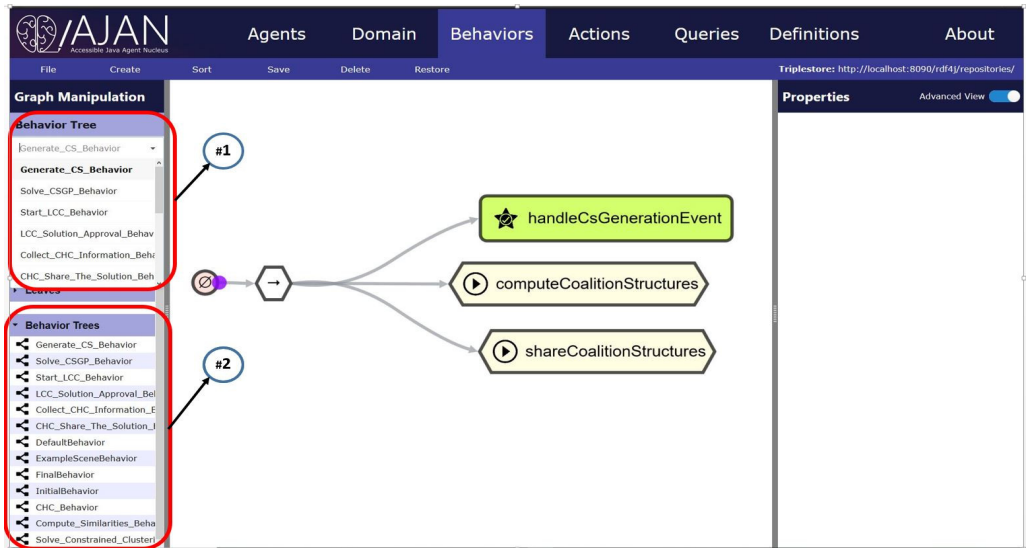

Figure A.2: Representation of template BTs in AJAN Editor

1. Is this method (having template BTs to create MAC use-cases) good enough for you to implement a use-case in a generic way? If no, what would you like to be modified?
2. Would you like to add anything more?

Concept 5: Agent Creation in AJAN Editor

To create an agent in AJAN Editor, the user needs to open the **New Agent Instance** popup page by clicking on the **Create Instance** button in the **Instance** section of the **Agents** panel. In this popup page, the user needs to define a name for the agent, select an Agent Template, and provide some Initial Knowledge in the respective fields as shown in figure A.3. By clicking on the Confirm button, an agent with the given inputs will be created and listed in the **Instance Overview** section.

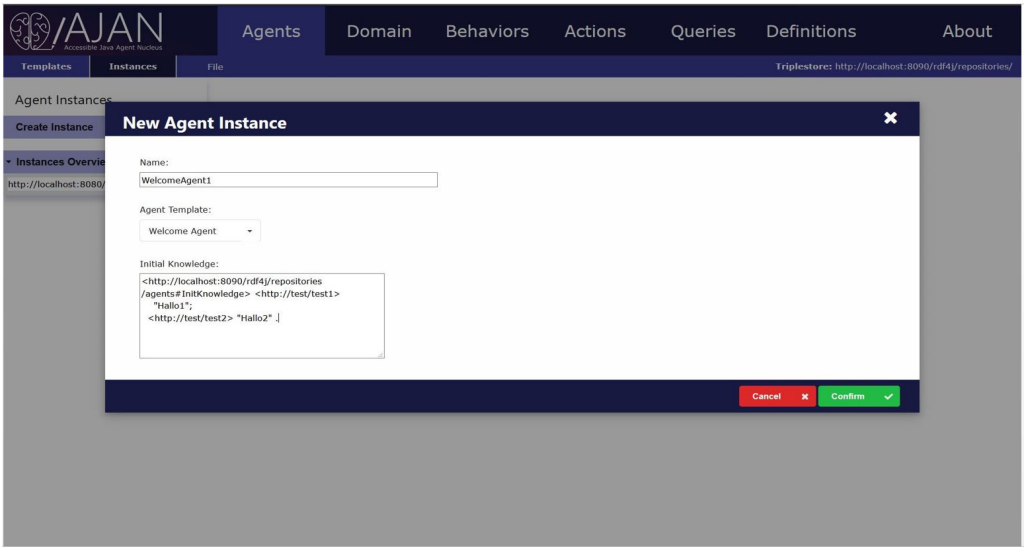

Figure A.3: Create Agent in AJAN Editor

1. Is this agent creation mechanism enough to create agents in the context of MAC? If no, what additional features would you like to have in terms of MAC?
2. Would you like to add anything more?

Concept 5: Start a MAC Process

Behaviors of AJAN agents are triggered via events. To trigger the behavior of an agent, it needs to receive a message to one of its endpoints. In the current version of AJAN Editor, users can send messages to the specific endpoints of agents. To do so, users need to follow the steps shown in figure A.4. Firstly, on the Instances Overview page, an agent instance should be selected (step 1 in figure A.4), then the list of endpoints of the selected agent is shown in the Messages to Agent Endpoints section and users can specify some agent knowledge as RDF in the area shown as step 2 in figure A.4. Finally, users need to click on the button in the circle near step 3 to send the message to the "Test Endpoint 2" endpoint of the selected agent.

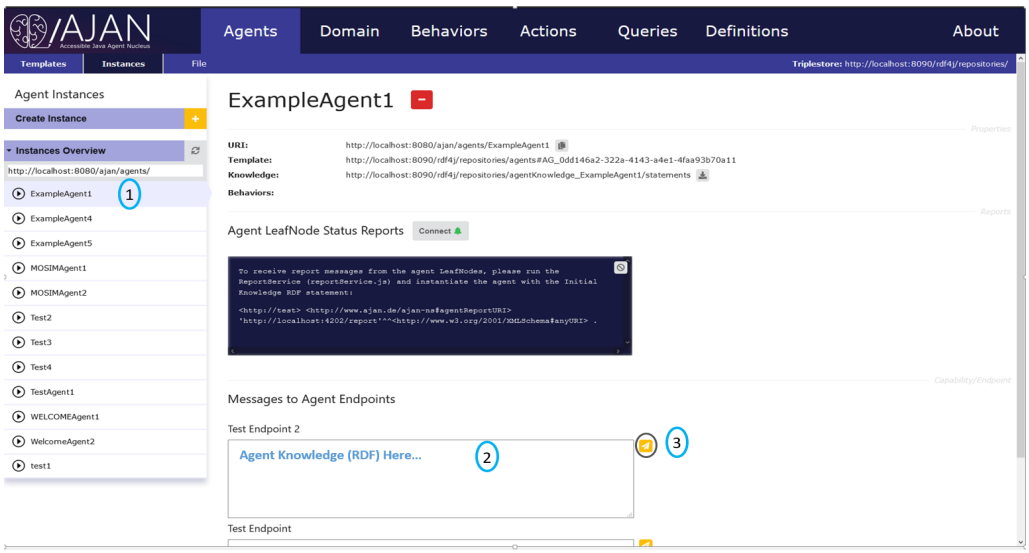

Figure A.4: Send a message to an endpoint of an agent in AJAN Editor

This feature makes it possible to send a message to an endpoint of an agent at a time. There is no feature to send the same message to the same endpoint of different agents at the same time in the current version of AJAN Editor. This feature might be necessary for different use-cases in which users might want to send a message to different agents in a MAS at the same time. Let's call this feature as One-to-Many feature, which means one message to many agents.

One-to-Many feature

This feature will be helpful when it is necessary to send a message to multiple agents at the same time in a MAC use-case. Users will need to do 3 actions:

1) Select agents,
2) Specify the message,
3) Select a common endpoint of the selected agents.

As described in the mock-up pictures of this feature, when users select a single agent (as shown in figure A.5), they will see the same UI as they see now in AJAN Editor (figure A.4). Figure A.6 describes the UI which will be displayed when users select multiple agents.

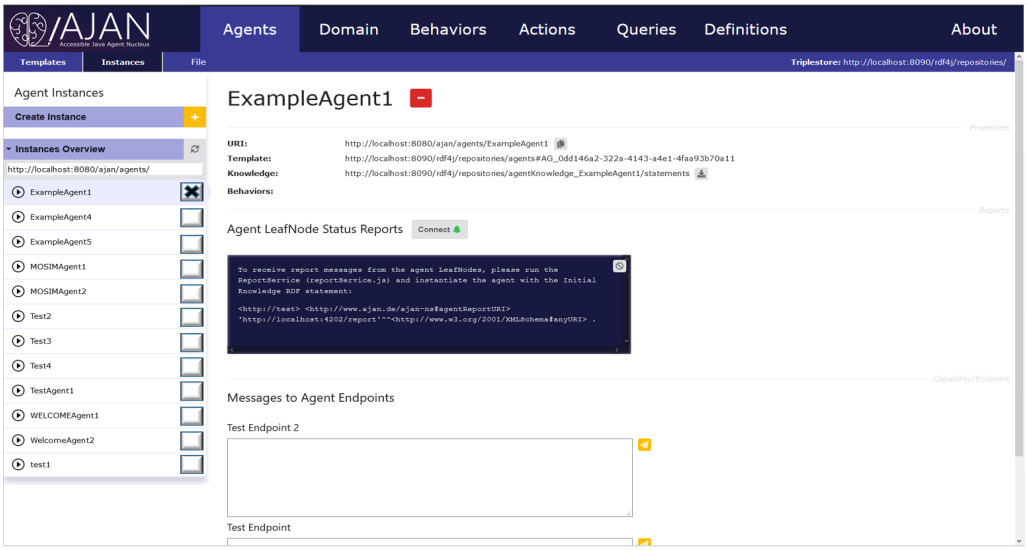

Figure A.5: Selection of a single agent

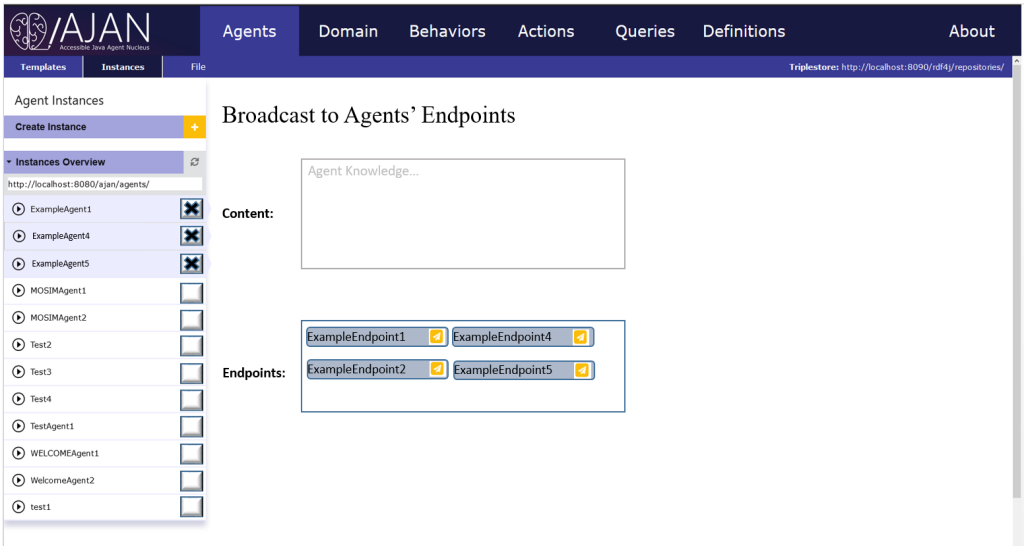

Figure A.6: Selection of multiple agents. One-to-Many feature

1. Figure A.4 describes the steps to send a message to an endpoint of an agent in AJAN Editor. If this feature is not sufficient to trigger the coordination behavior of an agent in a MAC use-case, in which way would you modify it in the context of MAC?
2. Would the concept of One-to-Many feature be useful for you? If so, in which context?
3. Is the design of One-to-Many feature (as shown in figure A.6) good enough for

you? If no, what would you like to be modified?

4. Would you like to add anything more?

A.3 Multiagent Coordination Use-Case Monitoring and Evaluation Questions

This section started with a description and then presented the concepts and their related questions in the questionnaire.

Description

MAJAN will provide the features to monitor the coordination process by logging the activities of agents and evaluate the solution of a MAC to see how well a distributed coordination is realized with SPARQL-BT-based AJAN agents. There will be 2 panels in MAJAN:
1) Monitoring and
2) Evaluation
This section asks questions regarding these two main features.

Concept 1: MAC Monitoring Panel

To be able to monitor the activities of agents, there will be a logger which displays the logs of coordinating agents as shown in figure A.7 - Multiagent Coordination Logs section. In other words, each coordinating agent will send logging information about its activities, and they will be shown to the user in the Monitoring Panel of MAJAN. With this feature, users will be able to monitor the whole coordination process. The entries listed below will be displayed as the log information: *agent's name, the title of the use-case, name of the algorithm, activity, and activity status*.
The activity field can be specified flexibly by the user by writing respective SPARQL queries, since different use-cases might have different workflows and therefore different activities might be important depending on the use-case. Activity status can be either "Succeeded", "Failed" or "Running".

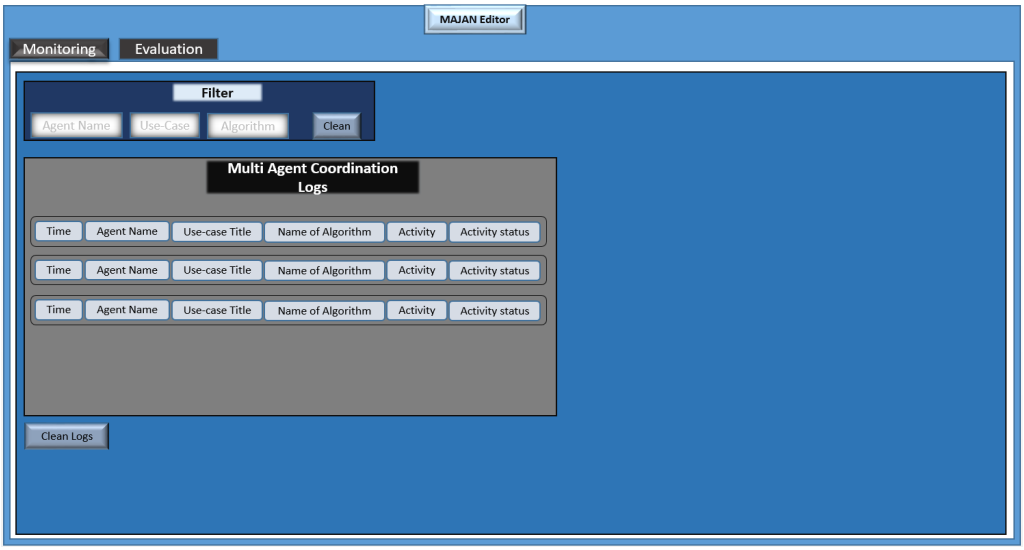

Figure A.7: Monitoring Panel

Additionally, users will be able to dynamically filter the logs based on the Agent Name, Use-Case, and Algorithm. Furthermore, the Clean button in the Filter section will clean the fields of the Filter section.

1. Is this feature useful and sufficient to monitor the activities of only coordinating agents? If not, what additional functionalities would you like to have in this section in terms of MAC?
2. Would you like to see any additional information (which is not already mentioned above) to be logged in the Multiagent Coordination Logs section? If so, which?
3. Would you like to add anything more?

Concept 2: MAC Solution Section in Evaluation Panel

This section will display the solution of a MAC process (figure A.8). Users will need to select the repository which contains the result(s). Then the results will be shown in the "Problem Instance" section such that if there are multiple results in the repository, users can select one of them to be displayed. In this section, the result will be described with the name of the use-case, its solver, and the problem ID. The problem ID represents the configuration data. For example, users can run the same use-case with the same solver multiple times for different data (input).

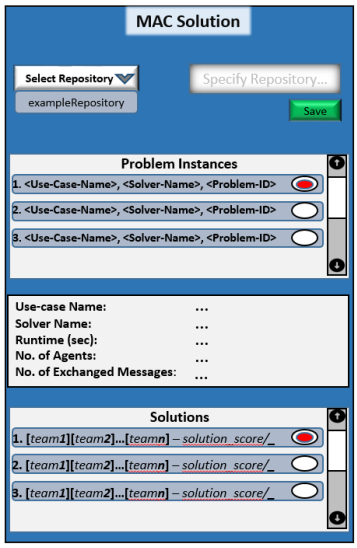

Figure A.8: MAC Solution section

An ontology to describe the result (of a MAC) will be created. If a user wants to evaluate his/her MAC result, then he/she must save the result in the predefined ontology such that MAJAN can understand and parse it. The following information about a MAC result will be displayed: title, number of agents, the runtime of the MAC process, and amount of exchanged messages.

Since agents might compute not only a single solution but a rank list of solutions, the "Solutions" section will display the produced solutions. To compare a MAC solution with a Baseline solution, users will need to select one of them in the "Solutions" section such that the selected one can be shown in the Comparison panel.

1. Is the technique of saving the MAC solution and displaying it as described in figure 13 good enough? If not, how would you like it to be modified?
2. Is there any other information that you would like to view in the MAC Solution section? If so, which?
3. Would you like to add anything more about the MAC Solution section?

Concept 3: Baseline Solution Section in Evaluation Panel

In this section, MAJAN will provide the features to upload a jar file of the algorithm, run it and display its solution as shown in figure A.9. The user will need to do the followings:
1) Upload the jar file of the centrally running coordination algorithm,
2) Enter the input for the algorithm as free text,
3) (Optional) User can save the input by giving a title and clicking on the Save button. The saved configurations will be displayed as a list in which the user can select one of them.
4) Click Run
5) In the background, MAJAN will execute the algorithm with the specified input, and it will display the solution once it is ready.

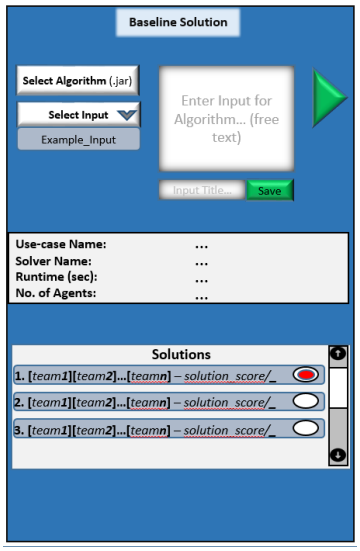

Figure A.9: Baseline Solution section

For these features to work successfully, there are 2 assumptions:
i) The user should enter the input in a correct way such that the algorithm is able to accept it, and
ii) The algorithm should be able to accept 2 arguments: 1. Input path and 2. Output path, to read the input and to save the output, respectively.
Additionally, all the information except the amount of exchanged messages, listed in the MAC Solution section will be displayed in the Baseline Solution section as well.

1. Are the features in the Baseline Solution section good enough to run a baseline algorithm and display its solution? If not, what would you like to be modified?
2. Is there any information, other than listed above, that you would like to view in the Baseline Solution section? If so, which?
3. Would you like to add anything more about the Baseline Solution section?

Concept 4: Comparison Section in Evaluation Panel

This section displays multiple metrics for comparison of the selected MAC and Baseline solutions. Furthermore, in the UI, there will be a small button (e.g. a question mark) near each metric which will show the description of the metric. The metrics for comparison are listed below, and they will be described in figure A.10:
- Total Runtime of Solutions
- Normalized Mutual Information (NMI)
- Rand Index (RI)
- Adjusted Rand Index (ARI)

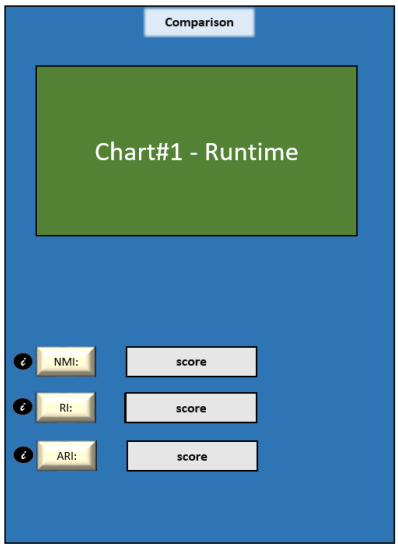

Figure A.10: Comparison section

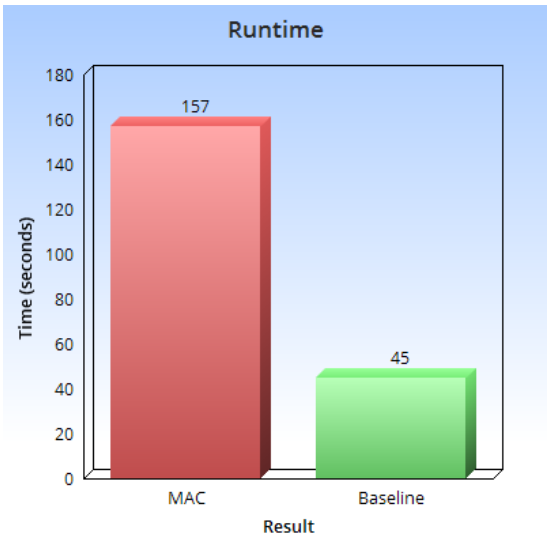

Figure A.11: Alternative Chart 1

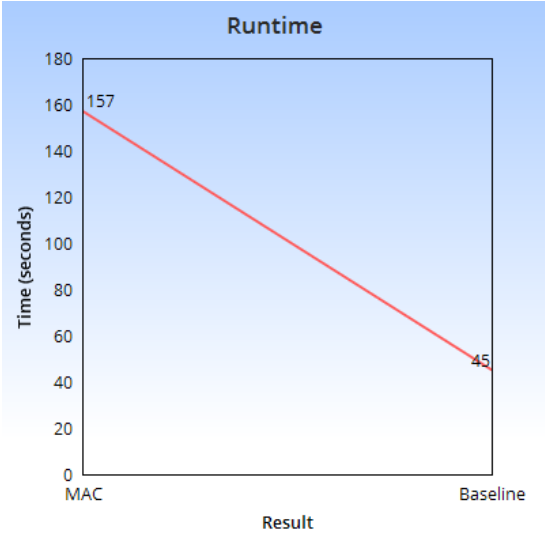

Figure A.12: Alternative Chart 2

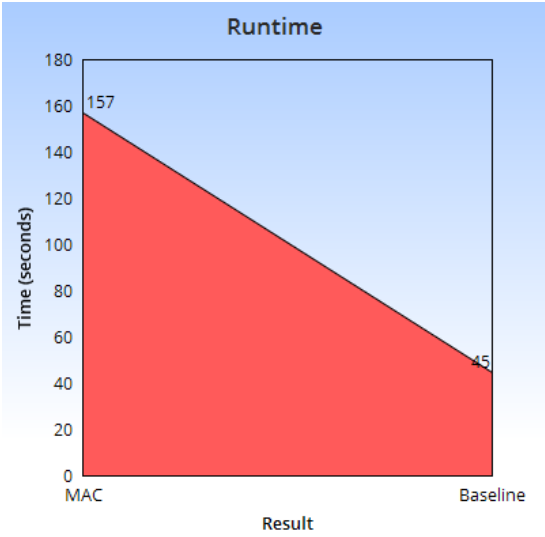

Figure A.13: Alternative Chart 3

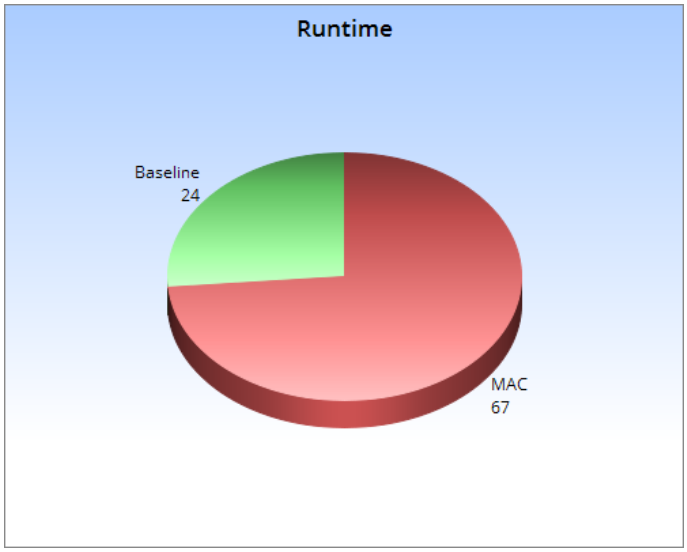

Figure A.14: Alternative Chart 4

1. Is there any metric other than Runtime, RI, ARI, and NMI that you would like to see in the Comparison section? If so, which?
2. Which of chart A.11, A.12, A.13 or A.14 do you like most for the representation of Runtime comparison? If none of them, do you have any specific preference in terms of Runtime comparison? (https://datavizproject.com/ provides some options)
3. Would you like to add anything more?

A.4 MAJAN Guideline Questions

In order to guide users in the process of realizing a MAC use-case with MAJAN, "MAJAN Guideline document" will be created. The document will be provided as a PDF file, and its structure will follow the workflow that is described in figure A.15.

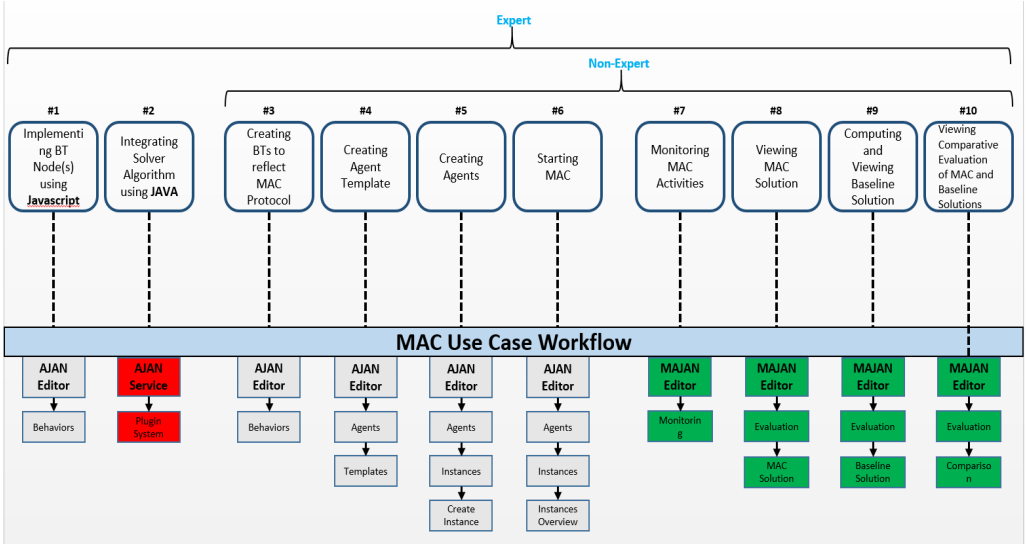

Figure A.15: Workflow for the Expert and Non-expert users to realize MAC Use-Cases

Basically, each task (from #1 to #10) in the workflow will be explained in a clear language together with the necessary screenshots of the components that are required to perform the task. The guideline document will include the instructions to create generic BT Nodes(s), to integrate an algorithm via Plugin System for MAC, to start MAC, to integrate a baseline algorithm, to compute a baseline solution, and so on.

1. If the structure of the MAJAN Guideline document (described above) is not clear enough, how would you like it to be modified?
2. Would you like to add anything more?

Appendix B
MAJAN Usability Evaluation Survey

This chapter presents all the questions and tasks that were asked in the MAJAN Evaluation Survey. The survey started with giving an introduction to the participants about the objective of MAJAN. Then they proceeded to complete the following sections:

B.1 User Background Questions

The participants answered the following background questions on a scale from one (lowest) to 5 (highest):

1. How familiar are you with AJAN? In which context?
2. How familiar are you with SPARQL-BTs? In which context?
3. How familiar are you with FIPA Request Protocol? In which context?
4. How familiar are you with Coalition Structure Generation Problem (CSGP)? In which context?
5. How familiar are you with Clustering? In which context?
6. How familiar are you with Postman API Platform?

B.2 Tasks

In this section, the participants completed the following tasks by interacting with the system:

- Task 1. Find MAJAN Guideline document and get familiar with its features by checking Overview section.
- Task 2. Use MAJAN Guideline document to find "CSGP Coordination Protocol", get familiar with it, and answer the questions below:

  Question 1. What is the objective in CSGP?

1. To partition objects in a way that the objects in the same groups are more similar to each other than objects in other groups
2. To partition agents into groups in a way that maximizes the social welfare of the system
3. To provide recommendations based on the recorded information on the users' preferences
4. To predict the value of a dependent variable based on an independent variable

Question 2. What is a Coalition Structure in CSGP?

1. A matrix that consists of Distance scores
2. A set that consists of Utility values
3. A group that consists of agents
4. A grouping that consists of groups

- Task 3. Use MAJAN Guideline document to create 5 agents
- Task 4. Use MAJAN Guideline document to populate Local Agents Repository (LAR) of created agents
- Task 5. Open MAJAN Monitoring UI in AJAN Editor and activate this feature
- Task 6. Use MAJAN Guideline document to start multiagent coordination for 5 agents and use "CSGP Coordination Protocol"
- Task 7. Open MAJAN Monitoring UI in AJAN Editor, check when the coordination is completed, and answer the question below

  Question 1. How to check the status of Multiagent coordination process by using Monitoring panel?

  1. There is no way to check
  2. By checking the "Status" column in Multiagent Coordination Logs
  3. By checking the "MAC Status" information in Activity column in Multiagent Coordination Logs

- Task 8. Open MAJAN Evaluation UI in AJAN Editor
- Task 9. Display the result of CSGP-Coordination-Protocol in MAC Solution section of MAJAN Evaluation UI
- Task 10. Check the features of Central Solution panel and answer the question below

  Question 1. Which of the following is not provided in Central Solution section?

  1. Executing a centrally running grouping algorithm
  2. Uploading and displaying a configuration of an algorithm
  3. Executing JAVA (*.jar) programs
  4. Executing Python (*.py) programs

- Task 11. Upload BOSS algorithm to Central Solution section and check its configuration info

- Task 12. Add input for BOSS algorithm to run 5 agents
- Task 13. Run BOSS algorithm and check its result in Central Solution section
- Task 14. Use Compare Solutions section in MAJAN UI to compare MAC and Central solutions
- Task 15. Compute Normalized Mutual Information (NMI) score of selected solutions

B.3 System Usability Scale Questions

Immediately after completing the tasks, the participants were asked to answer the System Usability Scale (SUS) questions [22] on a scale from one (strongly disagree) to five (strongly agree). The questions and the average of the rates that participants provided for each question are listed below [22]:

1. I think I would like to use this system frequently.
   Average Rate: 4 out of 5
2. I found the system unnecessarily complex.
   Average Rate: 1.4 out of 5
3. I thought the system was easy to use.
   Average Rate: 4.4 out of 5
4. I think that I would need the support of a technical person to be able to use this system.
   Average Rate: 1.4 out of 5
5. I found the various functions in this system were well integrated.
   Average Rate: 4.6 out of 5
6. I thought there was too much inconsistency in this system.
   Average Rate: 1 out of 5
7. I would imagine that most people would learn to use this system very frequently.
   Average Rate: 4.8 out of 5
8. I found the system very cumbersome to use.
   Average Rate: 2 out of 5
9. I felt very confident using the system.
   Average Rate: 3.6 out of 5
10. I needed to learn a lot of things before I could get going with the system.
    Average Rate: 2.2 out of 5

B.4 Complementary Questions

Finally, the survey was concluded with the additional questions to collect more feedback:

1. How satisfied are you with the look and feel of this software?
2. How satisfied are you with the layout and menu structure of the system?

3. How satisfied are you with the completeness of the available functionalities?
4. Do you have any other thoughts on how to improve this software?

Appendix C
LCC Data Records And Results

This chapter provides the LCC data records and their solutions computed by agents.

C.1 Data Records

1. Data Record 1 in table 4.2
   Characteristics: The TCNs were balanced in terms of their gender and nationality attributes (4 male vs 6 female). They were more inclined to prefer mixed groups, and only 20% of TCNs missed the previous lesson. Females mostly preferred the same gender and the mixed nationality groups, and males mostly preferred the mixed gender and the same nationality groups.

2. Data Record 2 in table C.1
   Characteristics: 50% of the males did not mind the gender and nationality of their group, while the 50% of females preferred to be in the same gender and mixed nationality groups. Only 10% of TCNs missed the previous lesson. 30%, 50% and 20% of TCNs had a CPL score in between 50 and 70, 70 and 90, and over 90, respectively.

Agent ID	Gender	Nationality	CPL	Attendance	Gender Preference	Nationality Preference
1	Male	C	72	Yes	Mixed	Same
2	Male	E	67	Yes	Same	Mixed
3	Female	D	85	Yes	Same	Same
4	Male	A	78	Yes	Mixed	Same
5	Female	B	66	Yes	Same	Mixed
6	Male	D	80	Yes	Don't mind	Don't mind
7	Female	E	75	Yes	Mixed	Don't mind
8	Male	B	57	Yes	Don't mind	Don't mind
9	Female	A	90	No	Don't mind	Mixed
10	Male	C	92	Yes	Don't mind	Don't mind

Table C.1: LCC data record 2

3. Data Record 3 in table C.2
   Characteristics: The TCNs were in complete balance in terms of their gender (i.e. 5 males and 5 females). Their nationalities were also in balance such that they were divided to 30%, 30% and 40% to belong to 3 nationalities. Females did not really care about the nationalities, but 80% of them preferred a same gender group. On the other hand, only 20% of males preferred the same gender and the rest were balanced (40% same vs 40% don't mind) in their gender preference. 30% of TCNs missed the previous lesson.

Agent ID	Gender	Nationality	CPL	Attendance	Gender Preference	Nationality Preference
1	Female	C	51	No	Same	Mixed
2	Male	B	83	Yes	Same	Mixed
3	Female	C	45	Yes	Same	Mixed
4	Male	A	74	Yes	Don't mind	Same
5	Female	C	53	Yes	Same	Don't mind
6	Male	B	73	No	Same	Don't mind
7	Female	B	63	Yes	Same	Don't mind
8	Male	A	70	Yes	Mixed	Same
9	Female	A	57	No	Mixed	Same
10	Male	C	81	Yes	Don't mind	Don't mind

Table C.2: LCC data record 3

4. Data Record 4 in table C.3
   Characteristics: There were 5 male and 5 females. Same, mixed and don't mind gender and nationality preferences were divided balanced as 30%, 30% and 40% respectively. Also, 30% of the TCNs missed the previous lesson. Moreover, 40%, 30%, 20% and 10% of the TCNs belonged to 4 different nationalities.

Agent ID	Gender	Nationality	CPL	Attendance	Gender Preference	Nationality Preference
1	Female	A	74	Yes	Mixed	Same
2	Male	C	84	No	Same	Mixed
3	Female	A	55	Yes	Same	Same
4	Male	B	93	Yes	Mixed	Same
5	Female	C	62	No	Same	Mixed
6	Male	A	58	Yes	Don't mind	Don't mind
7	Female	A	92	Yes	Mixed	Don't mind
8	Male	C	80	No	Don't mind	Don't mind
9	Female	D	70	Yes	Don't mind	Mixed
10	Male	B	66	Yes	Don't mind	Don't mind

Table C.3: LCC data record 4

5. Data Record 5 in table C.4
   Characteristics: 70% of the TCNs were female, and only females preferred the same gender (40%) group in the classroom. 40% and 70% of the TCNs did not mind the gender and nationality, respectively. And 30% of the TCNs preferred the same nationality. 50% of the TCNs had a CPL in between 50 and 70, 30% in between 70 and 90 and the rest in between 90 and 100 CPL.

Agent ID	Gender	Nationality	CPL	Attendance	Gender Preference	Nationality Preference
1	Male	B	88	Yes	Don't mind	Same
2	Female	D	66	Yes	Same	Same
3	Female	A	55	No	Mixed	Don't mind
4	Male	C	77	Yes	Don't mind	Don't mind
5	Female	B	51	No	Same	Same
6	Female	A	95	Yes	Mixed	Don't mind
7	Female	C	80	Yes	Same	Don't mind
8	Male	C	57	No	Don't mind	Don't mind
9	Female	A	90	Yes	Don't mind	Don't mind
10	Female	B	68	Yes	Same	Don't mind

Table C.4: LCC data record 5

6. Data Record 6 in table C.5
   Characteristics: The TCNs were in balance in terms of their gender (60% female, 40% male). 40%, 26%, 20% and 14% of them belong to different nationalities. Most of the females preferred the same gender and the mixed nationality groups. Most of the males preferred the same nationality and the mixed gender groups. And 26% of TCNs missed the previous lesson.

Agent ID	Gender	Nationality	CPL	Attendance	Gender Preference	Nationality Preference
1	Male	A	72	No	Don't mind	Same
2	Male	A	84	Yes	Mixed	Same
3	Female	B	73	Yes	Same	Mixed
4	Male	C	53	Yes	Mixed	Mixed
5	Female	D	25	Yes	Don't mind	Mixed
6	Female	B	44	Yes	Same	Mixed
7	Female	C	89	Yes	Mixed	Don't mind
8	Male	C	57	Yes	Mixed	Mixed
9	Female	D	95	Yes	Same	Mixed
10	Male	C	60	No	Mixed	Same
11	Female	A	74	Yes	Mixed	Same
12	Male	C	84	No	Same	Mixed
13	Female	A	55	Yes	Same	Same
14	Male	B	93	Yes	Mixed	Same
15	Female	C	62	No	Same	Mixed

Table C.5: LCC data record 6

7. Data Record 7 in table C.6
   Characteristics: 47% and 53% of the TCNs were females and males, respectively. Only 13% of TCNs missed the previous lesson. The TCNs who preferred the same gender or the nationality groups were divided in a balanced way (54% female vs 46% male). 47% of TCNs had a CPL in between 70 and 90, 33% were in between 50-70 and 7% had less CPL than 50.

Agent ID	Gender	Nationality	CPL	Attendance	Gender Preference	Nationality Preference
1	Male	C	72	Yes	Mixed	Same
2	Male	E	67	Yes	Same	Mixed
3	Female	D	84	Yes	Same	Same
4	Male	A	78	Yes	Mixed	Same
5	Female	B	66	Yes	Same	Mixed
6	Male	D	80	Yes	Don't mind	Don't mind
7	Female	E	75	Yes	Mixed	Don't mind
8	Male	B	57	No	Don't mind	Don't mind
9	Female	A	90	Yes	Don't mind	Mixed
10	Male	C	92	Yes	Don't mind	Don't mind
11	Female	C	51	No	Same	Mixed
12	Male	B	83	Yes	Same	Mixed
13	Female	C	45	Yes	Same	Mixed
14	Male	A	74	Yes	Don't mind	Same
15	Female	C	53	Yes	Same	Don't mind

Table C.6: LCC data record 7

8. Data Record 8 in table C.7
   Characteristics: In this data record, 53% of TCNs were females. Furthermore, 34% of TCNs had missed the previous lesson. Most of the TCNs preferred either the same or they did not care about the gender and nationality of the groups. Only 20% of them preferred the mixed gender or nationality groups. Additionally, the amount of TCNs who had a CPL in between 50 to 70 and 70 to 90 was the same and only 1 TCN had less CPL than 50.

Agent ID	Gender	Nationality	CPL	Attendance	Gender Preference	Nationality Preference
1	Female	C	51	No	Same	Mixed
2	Male	B	83	Yes	Same	Mixed
3	Female	C	45	Yes	Same	Mixed
4	Male	A	74	Yes	Don't mind	Same
5	Female	C	53	Yes	Same	Don't mind
6	Male	B	73	No	Same	Don't mind
7	Female	B	63	Yes	Same	Don't mind
8	Male	A	70	Yes	Mixed	Same
9	Female	A	57	No	Mixed	Same
10	Male	C	81	Yes	Don't mind	Don't mind
11	Male	B	88	Yes	Don't mind	Same
12	Female	D	66	Yes	Same	Same
13	Female	A	55	No	Mixed	Don't mind
14	Male	C	77	Yes	Don't mind	Don't mind
15	Female	B	51	No	Same	Same

Table C.7: LCC data record 8

9. Data Record 9 in table C.8
   Characteristics: 45% of TCNs were male. The lowest CPL of a TCN was 25 while the highest was 95, and they were divided in this range in a balanced way. In this example, TCNs mostly preferred the mixed groups. And 25% of them missed the

previous lesson. 84% and 77% of TCNs who preferred the same gender and the mixed nationality were females, respectively.

Agent ID	Gender	Nationality	CPL	Attendance	Gender Preference	Nationality Preference
1	Male	A	72	No	Don't mind	Same
2	Male	A	84	Yes	Mixed	Same
3	Female	B	73	Yes	Same	Mixed
4	Male	C	53	Yes	Mixed	Mixed
5	Female	D	25	Yes	Don't mind	Mixed
6	Female	B	44	Yes	Same	Mixed
7	Female	C	89	Yes	Mixed	Don't mind
8	Female	C	57	Yes	Mixed	Mixed
9	Female	D	95	Yes	Same	Mixed
10	Male	C	60	No	Mixed	Same
11	Female	A	74	Yes	Mixed	Same
12	Male	C	84	No	Same	Mixed
13	Female	A	55	Yes	Same	Same
14	Male	B	93	Yes	Mixed	Same
15	Female	C	62	No	Same	Mixed
16	Male	A	58	Yes	Don't mind	Don't mind
17	Female	A	92	Yes	Mixed	Don't mind
18	Male	C	80	No	Don't mind	Don't mind
19	Female	D	70	Yes	Don't mind	Mixed
20	Male	B	66	Yes	Don't mind	Don't mind

Table C.8: LCC data record 9

10. Data Record 10 in table C.9
    Characteristics: 45% of the TCNs were females and most of them preferred the same gender groups. But females were not so picky in their nationality preferences. On the other hand, males were divided equally in their gender preferences. When it comes to nationality preference, 40% of TCNs did not mind and the rest was divided in half as 30% preferred the same and 30% preferred the mixed nationalities.

Agent ID	Gender	Nationality	CPL	Attendance	Gender Preference	Nationality Preference
1	Male	C	72	Yes	Mixed	Same
2	Male	E	67	Yes	Same	Mixed
3	Female	D	85	Yes	Same	Same
4	Male	A	78	Yes	Mixed	Same
5	Female	B	66	Yes	Same	Mixed
6	Male	D	80	Yes	Don't mind	Don't mind
7	Female	E	75	Yes	Mixed	Don't mind
8	Male	B	57	No	Don't mind	Don't mind
9	Female	A	90	Yes	Don't mind	Mixed
10	Male	C	92	Yes	Don't mind	Don't mind
11	Female	C	51	No	Same	Mixed
12	Male	B	83	Yes	Same	Mixed
13	Female	C	45	Yes	Same	Mixed
14	Male	A	74	Yes	Don't mind	Same
15	Female	C	53	Yes	Same	Don't mind
16	Male	B	73	No	Same	Don't mind
17	Female	B	63	Yes	Same	Don't mind
18	Male	A	70	Yes	Mixed	Same
19	Female	A	57	No	Mixed	Same
20	Male	C	81	Yes	Don't mind	Don't mind

Table C.9: LCC data record 10

11. Data Record 11 in table C.10
    Characteristics: 60% of the TCNs were females and 67% of them preferred the same gender groups. 63% of males did not care about the gender of the group. Also, 55% of TCNs did not mind the nationality preference and 64% of them were females. Moreover, 30% of TCNs missed the previous lesson. Furthermore, 45% and 40% of TCNs had a CPL in between 50 to 70 and 70 to 90 respectively.

Agent ID	Gender	Nationality	CPL	Attendance	Gender Preference	Nationality Preference
1	Female	C	51	No	Same	Mixed
2	Male	C	83	Yes	Same	Mixed
3	Female	C	45	Yes	Same	Mixed
4	Male	A	74	Yes	Don't mind	Same
5	Female	C	53	Yes	Same	Don't mind
6	Male	B	73	No	Same	Don't mind
7	Female	B	63	Yes	Same	Don't mind
8	Male	A	70	Yes	Mixed	Same
9	Female	A	57	No	Mixed	Same
10	Male	C	81	Yes	Don't mind	Don't mind
11	Male	B	88	Yes	Don't mind	Same
12	Female	D	66	Yes	Same	Same
13	Female	A	55	No	Mixed	Don't mind
14	Male	C	77	Yes	Don't mind	Don't mind
15	Female	B	51	No	Same	Same
16	Female	A	95	Yes	Mixed	Don't mind
17	Female	C	80	Yes	Same	Don't mind
18	Male	C	57	No	Don't mind	Don't mind
19	Female	A	90	Yes	Don't mind	Don't mind
20	Female	B	68	Yes	Same	Don't mind

Table C.10: LCC data record 11

C.2 Results

The solutions that were computed by the agents through their coordination for the given data records and configurations are presented below. It is important to note that the numbers in the solutions represent the agent IDs. Moreover, the three best solutions that ranked from the best to the worst are displayed.

1. **Results of Data Record 1**
   (a) **Configuration 1**
      - **Total Computation time**: 141 seconds
      - **Solution List**:
         i. Rank 1: [[4, 8, 10], [5, 6], [2, 7, 9], [1, 3]]
         ii. Rank 2: [[4, 5, 6, 8, 10], [2, 7, 9], [1, 3]]
         iii. Rank 3: [[2, 7, 9], [4, 5, 6], [1, 3], [8, 10]]
      - **Customer Satisfaction Score**: 5

   (b) **Configuration 2**
      - **Total Computation time**: 138 seconds
      - **Solution List**:
         i. Rank 1: [[4, 8, 10], [5, 6], [1, 2, 7], [3, 9]]
         ii. Rank 2: [[1, 2, 4, 7], [3, 9], [5, 6], [8, 10]]

iii. Rank 3: [[2, 4, 8, 10], [1, 7], [3, 9], [5, 6]]

- **Customer Satisfaction Score**: 5

(c) **Configuration 3**

- **Total Computation time**: 132 seconds
- **Solution List**:
  i. Rank 1: [[2, 7, 9], [4, 8, 10], [1, 3], [5, 6]]
  ii. Rank 2: [[2, 7, 9], [4, 5, 6], [1, 3], [8, 10]]
  iii. Rank 3: [[1, 2, 3], [4, 8, 10], [5, 6], [7, 9]]
- **Customer Satisfaction Score**: 5

(d) **Configuration 4**

- **Total Computation time**: 143 seconds
- **Solution List**:
  i. Rank 1: [[4, 5, 6, 10], [1, 3, 8], [2, 7, 9]]
  ii. Rank 2: [[1, 2, 7, 9], [4, 5, 6], [3, 8, 10]]
  iii. Rank 3: [[5, 6, 8, 10], [1, 3, 4], [2, 7, 9]]
- **Customer Satisfaction Score**: 5

2. **Results of Data Record 2**

(a) **Configuration 1**

- **Total Computation time**: 140 seconds
- **Solution List**:
  i. Rank 1: [[1, 3, 4, 6, 7], [2, 5, 8], [9, 10]]
  ii. Rank 2: [[2, 5, 8], [3, 4, 6], [1, 7], [9, 10]]
  iii. Rank 3: [[1, 2, 5, 8], [3, 4, 6, 7], [9, 10]]
- **Customer Satisfaction Score**: 5

(b) **Configuration 2**

- **Total Computation time**: 137 seconds
- **Solution List**:
  i. Rank 1: [[1, 4, 6, 7], [2, 8], [3, 5], [9, 10]]
  ii. Rank 2: [[4, 6, 9, 10], [1, 7], [2, 8], [3, 5]]
  iii. Rank 3: [[1, 6, 9, 10], [2, 8], [3, 5], [4, 7]]
- **Customer Satisfaction Score**: 5

(c) **Configuration 3**

- **Total Computation time**: 140 seconds
- **Solution List**:

i. Rank 1: [[2, 5, 8], [3, 4, 6], [1, 7], [9, 10]]
ii. Rank 2: [[1, 4, 7], [2, 5, 8], [3, 6], [9, 10]]
iii. Rank 3: [[2, 5, 8], [3, 9, 10], [1, 7], [4, 6]]

- **Customer Satisfaction Score**: 5

(d) **Configuration 4**

- **Total Computation time**: 135 seconds
- **Solution List**:
  i. Rank 1: [[1, 2, 5, 8], [3, 9, 10], [4, 6, 7]]
  ii. Rank 2: [[1, 4, 6, 7], [2, 5, 8], [3, 9, 10]]
  iii. Rank 3: [[3, 6, 9, 10], [1, 4, 7], [2, 5, 8]]
- **Customer Satisfaction Score**: 5

3. **Results of Data Record 3**

(a) **Configuration 1**

- **Total Computation time**: 135 seconds
- **Solution List**:
  i. Rank 1: [[1, 3, 5], [4, 6, 8], [2, 10], [7, 9]]
  ii. Rank 2: [[1, 3, 5], [7, 8, 9], [2, 10], [4, 6]]
  iii. Rank 3: [[4, 6, 8], [5, 7, 9], [1, 3], [2, 10]]
- **Customer Satisfaction Score**: 5

(b) **Configuration 2**

- **Total Computation time**: 137 seconds
- **Solution List**:
  i. Rank 1: [[1, 3, 5, 7], [2, 6, 10], [4, 8, 9]]
  ii. Rank 2: [[1, 3, 5, 7], [2, 10], [4, 6], [8, 9]]
  iii. Rank 3: [[1, 3, 5, 7], [2, 4, 6, 10], [8, 9]]
- **Customer Satisfaction Score**: 5

(c) **Configuration 3**

- **Total Computation time**: 136 seconds
- **Solution List**:
  i. Rank 1: [[1, 3, 5], [4, 6, 8], [2, 10], [7, 9]]
  ii. Rank 2: [[4, 6, 8], [5, 7, 9], [1, 3], [2, 10]]
  iii. Rank 3: [[1, 3], [2, 10], [4, 6], [5, 9], [7, 8]]
- **Customer Satisfaction Score**: 5

(d) **Configuration 4**

- **Total Computation time**: 133 seconds
- **Solution List**:
  i. Rank 1: [[1, 3, 5, 7], [2, 6, 10], [4, 8, 9]]
  ii. Rank 2: [[1, 3, 5, 7], [2, 4, 6], [8, 9, 10]]
  iii. Rank 3: [[1, 3, 5, 7], [2, 8, 9], [4, 6, 10]]
- **Customer Satisfaction Score**: 5

4. **Results of Data Record 4**
   (a) **Configuration 1**
   - **Total Computation time**: 137 seconds
   - **Solution List**:
     i. Rank 1: [[1, 8, 9], [2, 4, 7], [3, 6], [5, 10]]
     ii. Rank 2: [[2, 4, 7], [5, 9, 10], [1, 8], [3, 6]]
     iii. Rank 3: [[1, 8], [9, 10], [2, 4, 7], [3, 5, 6]]
   - **Customer Satisfaction Score**: 5

   (b) **Configuration 2**
   - **Total Computation time**: 139 seconds
   - **Solution List**:
     i. Rank 1: [[1, 8, 9], [2, 6, 10], [3, 5], [4, 7]]
     ii. Rank 2: [[2, 6, 10], [3, 5, 9], [1, 8], [4, 7]]
     iii. Rank 3: [[1, 6, 8, 9], [2, 10], [3, 5], [4, 7]]
   - **Customer Satisfaction Score**: 5

   (c) **Configuration 3**
   - **Total Computation time**: 137 seconds
   - **Solution List**:
     i. Rank 1: [[1, 8, 9], [2, 4, 7], [3, 6], [5, 10]]
     ii. Rank 2: [[2, 4, 7], [5, 9, 10], [1, 8], [3, 6]]
     iii. Rank 3: [[2, 4, 7], [3, 5, 6], [1, 8], [9, 10]]
   - **Customer Satisfaction Score**: 5

   (d) **Configuration 4**
   - **Total Computation time**: 134 seconds
   - **Solution List**:
     i. Rank 1: [[1, 8, 9, 10], [2, 4, 7], [3, 5, 6]]
     ii. Rank 2: [[3, 5, 6, 10], [1, 8, 9], [2, 4, 7]]
     iii. Rank 3: [[3, 5, 6, 9], [1, 8, 10], [2, 4, 7]]

- **Customer Satisfaction Score**: 5

5. **Results of Data Record 5**
   (a) **Configuration 1**
      - **Total Computation time**: 140 seconds
      - **Solution List**:
        i. Rank 1: [[1, 6, 9], [3, 4, 7], [2, 8], [5, 10]]
        ii. Rank 2: [[1, 6, 9], [4, 7, 8], [2, 3], [5, 10]]
        iii. Rank 3: [[1, 6, 9], [3, 4, 7], [2, 5], [8, 10]]
      - **Customer Satisfaction Score**: 5

   (b) **Configuration 2**
      - **Total Computation time**: 136 seconds
      - **Solution List**:
        i. Rank 1: [[1, 3, 6, 9], [2, 5, 7, 10], [4, 8]]
        ii. Rank 2: [[1, 6, 8, 9], [2, 5, 7, 10], [3, 4]]
        iii. Rank 3: [[1, 3, 6, 9], [2, 7], [4, 8], [5, 10]]
      - **Customer Satisfaction Score**: 5

   (c) **Configuration 3**
      - **Total Computation time**: 157 seconds
      - **Solution List**:
        i. Rank 1: [[1, 6, 9], [3, 4, 7], [2, 8], [5, 10]]
        ii. Rank 2: [[1, 6, 9], [4, 7, 8], [2, 3], [5, 10]]
        iii. Rank 3: [[1, 6, 9], [3, 4, 7], [2, 5], [8, 10]]
      - **Customer Satisfaction Score**: 5

   (d) **Configuration 4**
      - **Total Computation time**: 143 seconds
      - **Solution List**:
        i. Rank 1: [[1, 3, 6, 9], [2, 5, 10], [4, 7, 8]]
        ii. Rank 2: [[1, 6, 8, 9], [2, 5, 10], [3, 4, 7]]
        iii. Rank 3: [[4, 6, 8, 9], [1, 5, 10], [2, 3, 7]]
      - **Customer Satisfaction Score**: 5

6. **Results of Data Record 6**
   (a) **Configuration 1**
      - **Total Computation time**: 209 seconds
      - **Solution List**:

i. Rank 1: [[4, 5, 6, 13, 15], [1, 3, 11], [7, 9, 14], [2, 12], [8, 10]]
ii. Rank 2: [[1, 3, 11], [4, 13, 15], [7, 9, 14], [2, 12], [5, 6], [8, 10]]
iii. Rank 3: [[1, 3, 11], [4, 8, 10], [7, 9, 14], [2, 12], [5, 6], [13, 15]]

- **Customer Satisfaction Score**: 5

(b) **Configuration 2**

- **Total Computation time**: 204 seconds
- **Solution List**:
  i. Rank 1: [[3, 6, 9, 13, 15], [4, 5, 8, 10], [1, 11], [2, 12], [7, 14]]
  ii. Rank 2: [[4, 8, 10], [1, 11], [2, 12], [3, 9], [5, 6], [7, 14], [13, 15]]
  iii. Rank 3: [[3, 9, 13, 15], [4, 8, 10], [1, 11], [2, 12], [5, 6], [7, 14]]
- **Customer Satisfaction Score**: 5

(c) **Configuration 3**

- **Total Computation time**: 201 seconds
- **Solution List**:
  i. Rank 1: [[1, 3, 11], [4, 13, 15], [7, 9, 14], [2, 12], [5, 6], [8, 10]]
  ii. Rank 2: [[1, 3, 11], [4, 8, 10], [7, 9, 14], [2, 12], [5, 6], [13, 15]]
  iii. Rank 3: [[1, 3, 11], [4, 10, 13], [7, 9, 14], [2, 12], [5, 6], [8, 15]]
- **Customer Satisfaction Score**: 5

(d) **Configuration 4**

- **Total Computation time**: 206 seconds
- **Solution List**:
  i. Rank 1: [[3, 5, 6, 15], [4, 8, 10, 13], [7, 9, 12, 14], [1, 2, 11]]
  ii. Rank 2: [[1, 11, 13], [2, 3, 12], [4, 8, 10], [5, 6, 15], [7, 9, 14]]
  iii. Rank 3: [[2, 3, 8, 15], [4, 5, 6, 10], [7, 9, 12, 14], [1, 11, 13]]
- **Customer Satisfaction Score**: 5

7. **Results of Data Record 7**

(a) **Configuration 1**

- **Total Computation time**: 250 seconds
- **Solution List**:
  i. Rank 1: [[1, 7, 14], [4, 6], [2, 5], [3, 12], [8, 15], [9, 10], [11, 13]]
  ii. Rank 2: [[1, 4, 6, 7, 14], [2, 5], [3, 12], [8, 15], [9, 10], [11, 13]]
  iii. Rank 3: [[1, 7, 14], [4, 6], [2, 5, 8], [11, 13, 15], [3, 12], [9, 10]]
- **Customer Satisfaction Score**: 5

(b) **Configuration 2**

- **Total Computation time**: 243 seconds
- **Solution List**:
  i. Rank 1: [[1, 4, 6, 7, 14], [5, 11, 13, 15], [2, 8], [3, 12], [9, 10]]
  ii. Rank 2: [[1, 4, 6, 7], [5, 11, 13, 15], [2, 8, 14], [3, 12], [9, 10]]
  iii. Rank 3: [[5, 11, 13, 15], [1, 4, 7], [2, 8], [3, 12], [6, 14], [9, 10]]
- **Customer Satisfaction Score**: 5

(c) **Configuration 3**

- **Total Computation time**: 239 seconds
- **Solution List**:
  i. Rank 1: [[1, 7, 14], [2, 5], [3, 12], [4, 6], [8, 15], [9, 10], [11, 13]]
  ii. Rank 2: [[1, 2, 5], [3, 12], [4, 6], [7, 14], [8, 15], [9, 10], [11, 13]]
  iii. Rank 3: [[1, 7, 14], [2, 5], [3, 12], [4, 6], [8, 13], [9, 10], [11, 15]]
- **Customer Satisfaction Score**: 5

(d) **Configuration 4**

- **Total Computation time**: 251 seconds
- **Solution List**:
  i. Rank 1: [[1, 4, 6, 7], [3, 9, 10, 12], [5, 11, 13, 15], [2, 8, 14]]
  ii. Rank 2: [[3, 9, 10, 12], [4, 6, 7, 14], [5, 11, 13, 15], [1, 2, 8]]
  iii. Rank 3: [[1, 2, 8, 14], [3, 9, 10, 12], [5, 11, 13, 15], [4, 6, 7]]
- **Customer Satisfaction Score**: 5

8. **Results of Data Record 8**

(a) **Configuration 1**

- **Total Computation time**: 201 seconds
- **Solution List**:
  i. Rank 1: [[2, 10, 11, 14], [8, 12, 13], [1, 5], [3, 15], [4, 6], [7, 9]]
  ii. Rank 2: [[2, 10, 11, 14], [8, 9, 12], [1, 5], [3, 15], [4, 6], [7, 13]]
  iii. Rank 3: [[2, 10, 11, 14], [8, 12, 13], [1, 3], [4, 6], [5, 15], [7, 9]]
- **Customer Satisfaction Score**: 5

(b) **Configuration 2**

- **Total Computation time**: 211 seconds
- **Solution List**:
  i. Rank 1: [[1, 3, 5, 12], [2, 10, 11], [4, 9], [6, 14], [7, 15], [8, 13]]
  ii. Rank 2: [[1, 5, 7, 12], [2, 10, 11], [3, 15], [4, 9], [6, 14], [8, 13]]
  iii. Rank 3: [[2, 6, 10, 11, 14], [1, 5, 7, 12], [3, 15], [4, 9], [8, 13]]

- **Customer Satisfaction Score**: 5

(c) **Configuration 3**

- **Total Computation time**: 207 seconds
- **Solution List**:
  i. Rank 1: [[2, 10, 11], [4, 6, 14], [8, 12, 13], [1, 5], [3, 15], [7, 9]]
  ii. Rank 2: [[2, 10, 11], [4, 6, 14], [8, 9, 12], [1, 5], [3, 15], [7, 13]]
  iii. Rank 3: [[2, 10, 11], [4, 6, 14], [8, 12, 13], [1, 3], [5, 15], [7, 9]]
- **Customer Satisfaction Score**: 5

(d) **Configuration 4**

- **Total Computation time**: 204 seconds
- **Solution List**:
  i. Rank 1: [[1, 7, 12], [2, 6, 11], [3, 5, 15], [4, 8, 9], [10, 13, 14]]
  ii. Rank 2: [[1, 7, 12], [2, 6, 10], [3, 5, 15], [4, 8, 9], [11, 13, 14]]
  iii. Rank 3: [[1, 7, 12], [2, 6, 14], [3, 5, 15], [4, 8, 9], [10, 11, 13]]
- **Customer Satisfaction Score**: 5

9. **Results of Data Record 9**

(a) **Configuration 1**

- **Total Computation time**: 577 seconds
- **Solution List**:
  i. Rank 1: [[7, 9, 14, 17], [3, 11, 18], [8, 10, 16], [1, 19], [2, 12], [4, 13], [5, 6], [15, 20]]
  ii. Rank 2: [[7, 9, 14, 17], [1, 3, 19], [8, 10, 16], [2, 12], [4, 13], [5, 6], [11, 18], [15, 20]]
  iii. Rank 3: [[7, 9, 14, 17], [3, 11, 18], [4, 8, 13], [1, 19], [2, 12], [5, 6], [10, 16], [15, 20]]
- **Customer Satisfaction Score**: 5

(b) **Configuration 2**

- **Total Computation time**: 560 seconds
- **Solution List**:
  i. Rank 1: [[3, 9, 13, 15, 19], [4, 8, 10, 16], [2, 7, 18], [1, 11], [5, 6], [12, 20], [14, 17]]
  ii. Rank 2: [[2, 7, 14, 17, 18], [3, 9, 13, 15, 19], [4, 8, 10, 16], [1, 11], [5, 6], [12, 20]]
  iii. Rank 3: [[4, 8, 10, 16], [2, 7, 18], [3, 9, 19], [1, 11], [5, 6], [12, 20], [13, 15], [14, 17]]
- **Customer Satisfaction Score**: 5

(c) **Configuration 3**

- **Total Computation time**: 581 seconds
- **Solution List**:
  i. Rank 1: [[8, 10, 16], [9, 14, 17], [1, 11], [2, 12], [3, 19], [4, 13], [5, 6], [7, 18], [15, 20]]
  ii. Rank 2: [[3, 11, 18], [8, 10, 16], [1, 19], [2, 12], [4, 13], [5, 6], [7, 17], [9, 14], [15, 20]]
  iii. Rank 3: [[1, 3, 19], [8, 10, 16], [2, 12], [4, 13], [5, 6], [7, 17], [9, 14], [11, 18], [15, 20]]
- **Customer Satisfaction Score**: 5

(d) **Configuration 4**

- **Total Computation time**: 570 seconds
- **Solution List**:
  i. Rank 1: [[3, 18, 19, 20], [4, 8, 10, 16], [1, 11, 13], [2, 7, 12], [5, 6, 15], [9, 14, 17]]
  ii. Rank 2: [[3, 18, 19, 20], [4, 5, 6, 15], [1, 11, 13], [2, 7, 12], [8, 10, 16], [9, 14, 17]]
  iii. Rank 3: [[3, 18, 19, 20], [4, 5, 6, 10], [1, 11, 13], [2, 7, 12], [8, 15, 16], [9, 14, 17]]
- **Customer Satisfaction Score**: 5

10. **Results of Data Record 10**

(a) **Configuration 1**

- **Total Computation time**: 636 seconds
- **Solution List**:
  i. Rank 1: [[1, 7, 14, 16, 18], [4, 6, 20], [2, 5], [3, 12], [8, 15], [9, 10], [11, 13], [17, 19]]
  ii. Rank 2: [[1, 4, 7, 18], [2, 5], [3, 12], [6, 20], [8, 15], [9, 10], [11, 13], [14, 16], [17, 19]]
  iii. Rank 3: [[1, 2, 5, 18], [3, 12], [4, 7], [6, 20], [8, 15], [9, 10], [11, 13], [14, 16], [17, 19]]
- **Customer Satisfaction Score**: 5

(b) **Configuration 2**

- **Total Computation time**: 601 seconds
- **Solution List**:
  i. Rank 1: [[5, 11, 13, 15, 17], [4, 14, 18, 19], [6, 16, 20], [1, 7], [2, 8], [3, 12], [9, 10]]
  ii. Rank 2: [[5, 11, 13, 15, 17], [2, 16], [6, 20], [4, 14, 18, 19], [1, 7, 8], [3, 12], [9, 10]]

iii. Rank 3: [[5, 11, 13, 15, 17], [4, 14, 18, 19], [1, 7, 20], [2, 8], [3, 12], [6, 16], [9, 10]]

- **Customer Satisfaction Score**: 5

(c) **Configuration 3**

- **Total Computation time**: 617 seconds
- **Solution List**:

i. Rank 1: [[1, 16, 18], [4, 6, 20], [2, 5], [3, 12], [7, 14], [8, 15], [9, 10], [11, 13], [17, 19]]

ii. Rank 2: [[1, 7, 18], [4, 6, 20], [2, 5], [3, 12], [8, 15], [9, 10], [11, 13], [14, 16], [17, 19]]

iii. Rank 3: [[4, 6, 20], [5, 17, 19], [1, 16], [2, 18], [3, 12], [7, 14], [8, 15], [9, 10], [11, 13]]

- **Customer Satisfaction Score**: 5

(d) **Configuration 4**

- **Total Computation time**: 627 seconds
- **Solution List**:

i. Rank 1: [[4, 14, 18, 19], [11, 13, 15, 17], [1, 7, 16], [2, 5, 8], [3, 9, 10], [6, 12, 20]]

ii. Rank 2: [[4, 14, 18, 19], [5, 11, 13, 15], [1, 7, 16], [2, 8, 17], [3, 9, 10], [6, 12, 20]]

iii. Rank 3: [[4, 14, 18, 19], [11, 13, 15, 17], [1, 7, 16], [2, 5, 8], [3, 9, 12], [6, 10, 20]]

- **Customer Satisfaction Score**: 5

11. **Results of Data Record 11**

(a) **Configuration 1**

- **Total Computation time**: 756 seconds
- **Solution List**:

i. Rank 1: [[2, 10, 14, 17], [8, 13, 20], [11, 16, 19], [1, 5], [3, 15], [4, 6], [7, 9], [12, 18]]

ii. Rank 2: [[2, 10, 14, 17], [8, 9, 20], [11, 16, 19], [1, 5], [3, 15], [4, 6], [7, 13], [12, 18]]

iii. Rank 3: [[2, 10, 14, 17], [8, 13, 20], [11, 16, 19], [1, 3], [4, 6], [5, 15], [7, 9], [12, 18]]

- **Customer Satisfaction Score**: 5

(b) **Configuration 2**

- **Total Computation time**: 708 seconds
- **Solution List**:

i. Rank 1: [[1, 7, 12, 17, 20], [2, 10, 18], [3, 5, 15], [11, 16, 19], [4, 9], [6, 14], [8, 13]]

ii. Rank 2: [[1, 7, 12, 17, 20], [2, 6, 10], [3, 5, 15], [11, 16, 19], [4, 9], [8, 13], [14, 18]]

iii. Rank 3: [[1, 7, 12, 17, 20], [10, 11, 16, 18, 19], [2, 6, 14], [3, 5, 15], [4, 9], [8, 13]]

- **Customer Satisfaction Score**: 5

(c) **Configuration 3**

- **Total Computation time**: 940 seconds
- **Solution List**:

i. Rank 1: [[8, 13, 20], [11, 16, 19], [1, 5], [2, 10], [3, 15], [4, 6], [7, 9], [12, 18], [14, 17]]

ii. Rank 2: [[8, 9, 20], [11, 16, 19], [1, 5], [2, 10], [3, 15], [4, 6], [7, 13], [12, 18], [14, 17]]

iii. Rank 3: [[2, 10, 17], [4, 6, 14], [8, 13, 20], [11, 16, 19], [1, 5], [3, 15], [7, 9], [12, 18]]

- **Customer Satisfaction Score**: 5

(d) **Configuration 4**

- **Total Computation time**: 657 seconds
- **Solution List**:

i. Rank 1: [[2, 6, 10, 14], [11, 16, 18, 19], [1, 12, 17], [3, 5, 13], [4, 8, 9], [7, 15, 20]]

ii. Rank 2: [[2, 6, 10, 14], [11, 16, 18, 19], [1, 17, 20], [3, 5, 15], [4, 8, 9], [7, 12, 13]]

iii. Rank 3: [[2, 6, 10, 14], [11, 16, 18, 19], [1, 7, 12], [3, 5, 15], [4, 8, 9], [13, 17, 20]]

- **Customer Satisfaction Score**: 5

Appendix D
CHC Data Records And Results

This chapter presents the CHC data records and their respective solutions.

D.1 Data Records

1. Data Record 1 is presented in tables 5.2 and 5.3 in section 5.4.
2. Data Record 2 in tables D.1 and D.2

ID	Age	Gender	Family	Nationality	Religion	Ethnicity
1	72	Female	Nuclear	Nat4	Rel1	Eth6
2	41	Other	Extended	Nat1	Rel5	Eth4
3	74	Other	Extended	Nat2	Rel3	Eth6
4	49	Female	Single Woman	Nat2	Rel1	Eth5
5	30	Female	Nuclear	Nat2	Rel7	Eth2
6	26	Male	Single Man	Nat2	Rel3	Eth3
7	21	Female	Extended	Nat2	Rel1	Eth7
8	29	Male	Single Man	Nat3	Rel3	Eth1
9	71	Male	Single Father	Nat7	Rel4	Eth3
10	63	Female	Nuclear	Nat3	Rel3	Eth5
11	18	Other	Extended	Nat5	Rel7	Eth2
12	45	Female	Extended	Nat1	Rel7	Eth4
13	36	Male	Extended	Nat5	Rel4	Eth7
14	63	Male	Extended	Nat4	Rel7	Eth6
15	46	Male	Single Father	Nat4	Rel3	Eth1
16	50	Other	Nuclear	Nat4	Rel5	Eth6
17	49	Male	Nuclear	Nat3	Rel6	Eth2
18	49	Male	Nuclear	Nat5	Rel3	Eth7
19	37	Female	Nuclear	Nat5	Rel4	Eth4
20	24	Female	Single Mother	Nat3	Rel3	Eth2

Table D.1: CHC data record 2: Demographic data of TCNs

ID	Age P.	Gender P.	Family P.	Nationality P.	Religion P.	Ethnicity P.	Location P.	Accessibility P.	Rent Period P.	Share-with P.
1	Don't mind	Female	Single Woman/Single Mother	Same	Mixed	Same	L1/L7	No	1/1/2021-1/7/2022	1
2	34-43	Don't mind	Single Mother	Don't mind	Same	Mixed	L3/L5	No	1/1/2021-1/7/2022	2-4
3	51-65/65-120	Male	Single Father/Single Man	Mixed	Same	Mixed	L3/L6	Don't mind	1/3/2021-1/7/2022	3
4	18-25/26-33/34-43	Don't mind	Don't mind	Don't mind	Same	Mixed	L6/L2	Don't mind	1/1/2021-1/7/2021	4
5	26-33	Don't mind	Single Father	Mixed	Mixed	Same	L5	Don't mind	1/1/2021-1/4/2021	5
6	34-43	Female	Single Mother	Same	Same	Dont mind	Don't mind	Don't mind	1/1/2021-1/7/2022	2-7
7	18-25/26-33	Don't mind	Extended	Don't mind	Mixed	Same	L5	No	1/5/2021-1/5/2022	2-7
8	34-43	Female	Single Mother/Single Father/Extended	Mixed	Don't mind	Don't mind	L4/L2/L7	Yes	1/1/2021-1/7/2022	1
9	Don't mind	Female	Single Mother	Same	Don't mind	Mixed	L1	Don't mind	15/3/2021-1/4/2022	4
10	Don't mind	Female	Single Woman/Single Mother	Same	Don't mind	Same	L6	No	1/1/2021-1/7/2021	4
11	Don't mind	Don't mind	Don't mind	Same	Don't mind	Mixed	L3	Don't mind	Don't mind	2-7
12	34-43/51-65	Don't mind	Nuclear	Mixed	Same	Mixed	L3	Yes	1/9/2021-1/12/2021	2-7
13	18-25/26-33	Female	Single Mother	Mixed	Mixed	Same	L4/L2/L7	No	1/1/2021-1/7/2021	3-5
14	Don't mind	Other	Don't mind	Same	Don't mind	Mixed	L4/L2/L7	Yes	1/3/2021-1/7/2022	1-2
15	44-50	Don't mind	Single Father	Mixed	Same	Don't mind	L7	Yes	1/9/2021-1/12/2021	2-3
16	65-120	Male	Nuclear	Don't mind	Same	Don't mind	Don't mind	Don't mind	Don't mind	1
17	34-43	Male	Single Father	Don't mind	Don't mind	Don't mind	L6	Don't mind	1/1/2021-1/7/2022	Don't mind
18	Don't mind	Other	Extended	Mixed	Same	Mixed	L3/L6/L7	No	1/1/2021-1/7/2022	1-5
19	34-43	Don't mind	Extended/Nuclear	Same	Mixed	Mixed	L7	Don't mind	Don't mind	1-5
20	18-25	Male	Extended/Nuclear	Don't mind	Mixed	Same	L1/L7	No	15/3/2021-1/4/2022	1-2

Table D.2: CHC data record 2: Individual Preferences of TCNs

3. Data Record 3 in tables D.3 and D.4

ID	Age	Gender	Family	Nationality	Religion	Ethnicity
1	65	Female	Extended	Nat5	Rel7	Eth3
2	37	Female	Single Mother	Nat2	Rel7	Eth1
3	58	Other	Extended	Nat2	Rel2	Eth2
4	40	Other	Nuclear	Nat3	Rel5	Eth7
5	34	Other	Single Father	Nat2	Rel6	Eth4
6	36	Male	Extended	Nat7	Rel5	Eth6
7	43	Male	Single Father	Nat5	Rel4	Eth4
8	41	Male	Single Man	Nat3	Rel5	Eth1
9	24	Female	Single Woman	Nat1	Rel3	Eth5
10	25	Female	Extended	Nat2	Rel7	Eth3
11	66	Female	Extended	Nat7	Rel6	Eth7
12	29	Other	Single Woman	Nat2	Rel5	Eth2
13	71	Other	Single Mother	Nat3	Rel7	Eth5
14	47	Male	Nuclear	Nat2	Rel3	Eth1
15	56	Female	Extended	Nat1	Rel2	Eth1
16	34	Female	Single Mother	Nat3	Rel1	Eth3
17	24	Other	Single Father	Nat2	Rel6	Eth5
18	37	Male	Single Father	Nat1	Rel7	Eth4
19	34	Other	Single Father	Nat1	Rel7	Eth4
20	41	Male	Nuclear	Nat6	Rel5	Eth6
21	31	Other	Extended	Nat4	Rel3	Eth5
22	25	Other	Nuclear	Nat4	Rel1	Eth5
23	56	Male	Extended	Nat4	Rel3	Eth6
24	74	Female	Single Mother	Nat5	Rel4	Eth3
25	49	Female	Single Woman	Nat7	Rel1	Eth2

Table D.3: CHC data record 3: Demographic data of TCNs

ID	Age P.	Gender P.	Family P.	Nationality P.	Religion P.	Ethnicity P.	Location P.	Accessibility P.	Rent Period P.	Share-with P.
1	26-33	Male	Extended	Mixed	Same	Same	L6/L2	Yes	1/1/2021-1/7/2021	3
2	Don't mind	Don't mind	Don't mind	Don't mind	Don't mind	Mixed	L3/L6	Yes	1/9/2021-1/12/2021	3
3	65-120	Female	Single Woman	Mixed	Mixed	Mixed	L5	Don't mind	1/9/2021-1/12/2021	1-4
4	18-25/26-33	Other	Don't mind	Don't mind	Don't mind	Mixed	Don't mind	No	1/9/2021-1/12/2021	1-3
5	44-50/51-65	Don't mind	Single Father	Don't mind	Don't mind	Mixed	L7/L6/L5	Don't mind	1/5/2021-1/5/2022	1-2
6	44-50/51-65	Don't mind	Extended	Same	Mixed	Same	L2	Yes	15/3/2021-1/4/2022	3
7	18-25/26-33/34-43	Female	Single Woman/Single Mother	Same	Same	Mixed	L2	No	3/7/2021-12/2/2022	1
8	34-43	Don't mind	Single Man	Don't mind	Don't mind	Same	L2/L3	Don't mind	1/1/2021-1/7/2021	3-4
9	18-25	Female	Don't mind	Mixed	Same	Mixed	L1	Yes	1/1/2021-1/7/2021	1-5
10	18-25/26-33	Female	Nuclear	Same	Don't mind	Mixed	L1/L2/L3	Yes	1/5/2021-1/5/2022	1
11	51-65/65-120	Male	Don't mind	Same	Mixed	Mixed	L5	Don't mind	1/1/2021-1/7/2022	5
12	18-25/26-33/34-43	Male	Single Father/Single Man	Don't mind	Same	Mixed	Don't mind	Don't mind	23/4/2020-17/5/2021	1-2
13	65-120	Other	Single Father/Single Man	Mixed	Don't mind	Same	L6/L2	No	1/5/2021-1/5/2022	3-5
14	Don't mind	Female	Nuclear	Mixed	Same	Mixed	L5	Yes	1/1/2021-1/4/2021	1-5
15	51-65/65-120	Female	Single Woman/Single Mother	Don't mind	Don't mind	Mixed	L3/L6	Don't mind	1/1/2021-1/4/2021	2-7
16	26-33/34-43	Don't mind	Don't mind	Mixed	Mixed	Mixed	L4	Yes	3/7/2021-12/2/2022	2-4
17	44-50/51-65	Don't mind	Single Mother/Single Father/Extended	Don't mind	Mixed	Mixed	L4	Don't mind	1/1/2021-1/7/2022	3-4
18	18-25	Male	Single Man	Same	Same	Don't mind	L4	Yes	1/5/2021-1/5/2022	Don't mind
19	51-65/65-120	Other	Extended	Mixed	Mixed	Don't mind	L3/L6	Yes	1/1/2021-1/7/2021	3
20	51-65	Female	Extended	Same	Mixed	Don't mind	L3/L5	Yes	1/1/2021-1/7/2022	5
21	51-65/65-120	Other	Extended/Nuclear	Same	Same	Don't mind	L7/L6/L5	No	Don't mind	1-3
22	Don't mind	Other	Single Woman/Single Mother	Mixed	Mixed	Mixed	L3	No	1/3/2021-1/7/2022	2-3
23	Don't mind	Male	Single Mother/Single Father/Extended	Mixed	Same	Mixed	L4/L2/L7	Don't mind	15/3/2021-1/4/2022	1-5
24	44-50/51-65	Female	Single Woman/Single Mother	Same	Same	Don't mind	Don't mind	Yes	1/3/2021-1/7/2022	2-7
25	18-25/26-33/34-43	Female	Single Woman/Single Mother	Don't mind	Mixed	Don't mind	L1	Don't mind	1/1/2021-1/4/2021	3-5

Table D.4: CHC data record 3: Individual Preferences of TCNs

4. Data Record 4 in tables D.5 and D.6

ID	Age	Gender	Family	Nationality	Religion	Ethnicity
1	56	Male	Single Father	Nat4	Rel5	Eth7
2	59	Other	Extended	Nat1	Rel2	Eth4
3	28	Female	Single Woman	Nat1	Rel2	Eth2
4	20	Male	Single Man	Nat6	Rel7	Eth3
5	66	Female	Nuclear	Nat5	Rel5	Eth3
6	64	Female	Extended	Nat5	Rel3	Eth6
7	59	Male	Single Father	Nat5	Rel3	Eth4
8	61	Male	Single Father	Nat5	Rel3	Eth7
9	22	Other	Single Woman	Nat6	Rel1	Eth4
10	35	Male	Extended	Nat1	Rel5	Eth2
11	72	Female	Extended	Nat1	Rel7	Eth2
12	48	Other	Single Father	Nat1	Rel5	Eth6
13	37	Female	Single Mother	Nat5	Rel3	Eth4
14	53	Male	Nuclear	Nat6	Rel5	Eth7
15	54	Male	Extended	Nat3	Rel2	Eth7
16	22	Female	Extended	Nat6	Rel2	Eth1
17	34	Other	Single Woman	Nat3	Rel3	Eth4
18	65	Other	Extended	Nat5	Rel4	Eth3
19	37	Female	Single Mother	Nat2	Rel2	Eth3
20	42	Male	Single Father	Nat1	Rel1	Eth6
21	20	Female	Single Mother	Nat1	Rel1	Eth1
22	40	Male	Single Man	Nat3	Rel2	Eth2
23	34	Male	Single Father	Nat2	Rel1	Eth3
24	19	Other	Extended	Nat1	Rel3	Eth2
25	67	Female	Single Mother	Nat1	Rel1	Eth1
26	43	Male	Single Man	Nat4	Rel4	Eth4
27	33	Female	Single Woman	Nat2	Rel3	Eth3
28	27	Female	Nuclear	Nat3	Rel2	Eth2
29	55	Male	Nuclear	Nat1	Rel4	Eth3
30	23	Other	Single Woman	Nat2	Rel5	Eth4
31	18	Female	Single Woman	Nat2	Rel5	Eth4
32	24	Female	Single Mother	Nat2	Rel5	Eth4
33	29	Male	Extended	Nat5	Rel3	Eth2
34	37	Male	Single Man	Nat1	Rel4	Eth3
35	21	Male	Nuclear	Nat4	Rel1	Eth6
36	69	Female	Single Woman	Nat7	Rel2	Eth1
37	31	Female	Extended	Nat2	Rel5	Eth1
38	26	Other	Single Woman	Nat5	Rel3	Eth5
39	37	Male	Nuclear	Nat1	Rel7	Eth2
40	20	Other	Single Woman	Nat3	Rel2	Eth1
41	22	Other	Single Mother	Nat7	Rel5	Eth3
42	33	Male	Nuclear	Nat4	Rel4	Eth3
43	44	Male	Nuclear	Nat5	Rel4	Eth4
44	55	Male	Extended	Nat1	Rel1	Eth5
45	66	Female	Single Woman	Nat3	Rel5	Eth3
46	77	Female	Single Mother	Nat4	Rel7	Eth3
47	50	Male	Nuclear	Nat5	Rel6	Eth7
48	36	Female	Extended	Nat7	Rel4	Eth2
49	41	Male	Nuclear	Nat1	Rel4	Eth3
50	25	Male	Single Man	Nat3	Rel2	Eth1

Table D.5: CHC data record 4: Demographic data of TCNs

ID	Age P.	Gender P.	Family P.	Nationality P.	Religion P.	Ethnicity P.	Location P.	Accessibility P.	Rent Period P.	Share-with P.
1	44-50/51-65	Don't mind	Don't mind	Mixed	Don't mind	Same	L1	Yes	1/3/2021-1/7/2022	2-3
2	65-120	Female	Single Mother/Single Father/Extended	Mixed	Mixed	Mixed	L2/L3	Yes	Don't mind	2-7
3	34-43	Other	Extended/Nuclear	Don't mind	Same	Mixed	L4/L2/L7	Don't mind	3/7/2021-12/2/2022	5
4	18-25	Male	Don't mind	Same	Same	Same	L4	No	1/1/2021-1/7/2021	3-5
5	34-43	Female	Single Woman/Single Mother	Mixed	Don't mind	Same	L1	Don't mind	1/1/2021-1/4/2021	1-3
6	Don't mind	Female	Single Mother/Single Father/Extended	Mixed	Same	Mixed	L2/L3	Yes	1/1/2021-1/7/2021	2-4
7	18-25/26-33/34-43	Male	Don't mind	Don't mind	Same	Same	L7/L6/L5	Don't mind	1/1/2021-1/4/2021	2-7
8	44-50	Don't mind	Single Man	Mixed	Same	Same	L6	Don't mind	1/1/2021-1/7/2021	1-2
9	Don't mind	Don't mind	Don't mind	Don't mind	Same	Mixed	L5	No	1/3/2021-1/7/2022	1-5
10	44-50/51-65	Don't mind	Don't mind	Don't mind	Mixed	Same	L3/L5	Don't mind	1/1/2021-1/7/2021	1-2
11	51-65/65-120	Female	Don't mind	Don't mind	Same	Don't mind	L5	Yes	23/4/2020-17/5/2021	Don't mind
12	51-65	Don't mind	Extended/Nuclear	Mixed	Don't mind	Don't mind	L4	No	1/1/2021-1/7/2022	2-4
13	44-50	Female	Nuclear	Mixed	Same	Don't mind	L6/L2	Don't mind	Don't mind	1-3
14	51-65/65-120	Male	Extended	Same	Mixed	Mixed	L4/L2/L7	Yes	15/3/2021-1/4/2022	2
15	18-25/26-33/34-43	Male	Extended/Nuclear	Same	Don't mind	Don't mind	L7/L6/L5	Yes	1/5/2021-1/5/2022	Don't mind
16	18-25/26-33/34-43	Female	Single Mother/Single Father/Extended	Mixed	Don't mind	Don't mind	L6	Yes	1/1/2021-1/7/2022	1-5
17	26-33/34-43/65-120	Male	Single Man	Mixed	Don't mind	Same	L2/L3	Yes	23/4/2020-17/5/2021	1
18	65-120	Male	Don't mind	Same	Mixed	Mixed	L7	Yes	1/1/2021-1/7/2021	1-2
19	51-65	Male	Single Mother/Single Father/Extended	Don't mind	Mixed	Same	L7/L6/L5	Don't mind	1/5/2021-1/5/2022	Don't mind
20	51-65	Male	Single Man	Don't mind	Mixed	Mixed	L1/L2/L3	No	1/3/2021-1/7/2022	5
21	18-25	Female	Single Woman/Single Mother	don't mind	don't mind	Mixed	don't mind	don't mind	1/1/2021-1/7/2021	2-3
22	don't mind	Mixed	don't mind	Mixed	don't mind	Same	L1/L2/L3	don't mind	1/5/2021-1/5/2022	Don't mind
23	26-33/34-43	Don't mind	Single Mother/Single Father/Nuclear	don't mind	Same	Don't mind	L3/L4/L5	Yes	1/1/2021-1/7/2022	2-4
24	18-25	Don't mind	Don't mind	don't mind	Same	Mixed	don't mind	don't mind	Don't mind	Don't mind
25	Don't mind	Female	Single Woman/Single Mother/Nuclear	Same	Same	Don't mind	don't mind	Yes	Don't mind	1-2
26	34-43/44-50	Don't mind	Don't mind	don't mind	don't mind	Mixed	don't mind	don't mind	1/3/2021-1/7/2022	1-3
27	18-25/26-33/34-43	Female	Single Woman/Single Mother	Same	don't mind	Mixed	L1/L2/L3/L4	No	1/1/2021-1/7/2022	Don't mind
28	Don't mind	Don't mind	Single Woman/Single Mother/Nuclear	Mixed	Mixed	Same	L3/L6/L9	Yes	Don't mind	1-4
29	51-65/65-120	Male	Don't mind	Same	Same	Same	L4	Yes	1/5/2021-1/12/2021	1-2
30	Don't mind	Female	Single Woman/Single Mother	don't mind	don't mind	Don't mind	don't mind	No	1/1/2021-1/12/2021	Don't mind
31	18-25	Female	Single Woman/Single Mother	Same	Same	Same	L7/L8	No	1/1/2021-1/4/2021	3
32	Don't mind	Don't mind	Single Woman/Single Mother/Nuclear	Same	Same	Same	L3/L6/L7/L9	Don't mind	Don't mind	Don't mind
33	26-33/34-43	Don't mind	Extended	Don't mind	Same	Don't mind	L1/L2/L3/L4	Yes 1/9/2021-1/2/2022	Don't mind	
34	Don't mind	Don't mind	Single Man/Single Father	Mixed	don't mind	Don't mind	don't mind	Don't mind	Don't mind	1-3
35	18-25	Mixed	Don't mind	Same	Mixed	Mixed	L3/L12	don't mind	1/4/2021-1/4/2022	2-3
36	51-65/65-120	Female	Don't mind	Same	Same	Same	Don't mind	Yes	1/7/2021-1/12/2021	Don't mind
37	18-25/26-33/34-43	Female	Single Woman/Single Mother/Extended	Don't mind	don't mind	don't mind	L4/L7/L12/L14	Don't mind	1/1/2021-1/7/2022	1-2
38	18-25/26-33	Other	Don't mind	Same	Don't mind	Mixed	don't mind	Don't mind	Don't mind	Don't mind
39	26-33/34-43	Don't mind	Nuclear	Same	Mixed	Same	L2/L3/L4	Yes	15/3/2022-15/10/2022	Don't mind
40	18-25	Female	Single Mother	Mixed	Mixed	Same	L1/L11	Yes	5/1/2021-5/7/2021	2-3
41	44-50	Don't mind	Single Woman	Same	Don't mind	Mixed	Don't mind	No	1/5/2021-1/5/2022	2
42	26-33/51-65/65-120	Male	Single Mother	Mixed	Don't mind	Mixed	L2	No	3/7/2021-12/2/2022	3-4
43	26-33	Other	Nuclear	Don't mind	Mixed	Mixed	L5	Don't mind	Don't mind	2
44	Don't mind	Don't mind	Don't mind	Don't mind	Same	Same	L6	Yes	1/1/2021-1/7/2022	1-4
45	44-50/51-65	Female	Nuclear	Same	Mixed	Mixed	L2/L3	Don't mind	1/1/2021-1/4/2021	1-3
46	Don't mind	Female	Single Mother/Extended/Nuclear	Don't mind	Don't mind	Same	L4/L6/L8	Yes	1/5/2021-1/12/2021	3-4
47	44-50/51-65	Other	Single Father	Same	Mixed	Mixed	L1/L2/L3	Don't mind	1/1/2021-1/7/2022	1-5
48	18-25/26-33/34-43	Male	Don't mind	Don't mind	Mixed	Don't mind	L3/L5	Don't mind	1/1/2021-1/7/2022	2-4
49	26-33/51-65/65-120	Don't mind	Don't mind	don't mind	don't mind	don't mind	don't mind	don't mind	1/3/2021-1/10/2021	2
50	18-25/26-33/34-43	Mixed	Don't mind	Mixed	Mixed	Mixed	don't mind	Don't mind	13/12/2021-1/5/2022	1-4

Table D.6: CHC data record 4: Individual Preferences of TCNs

5. Data Record 5 in tables D.7 and D.8

ID	Age	Gender	Family	Nationality	Religion	Ethnicity
1	39	Female	Extended	Nat7	Rel4	Eth3
2	37	Male	Single Man	Nat1	Rel1	Eth2
3	39	Male	Single Father	Nat7	Rel2	Eth3
4	52	Other	Single Father	Nat2	Rel7	Eth1
5	70	Other	Single Woman	Nat5	Rel4	Eth2
6	71	Other	Extended	Nat1	Rel2	Eth3
7	24	Male	Single Man	Nat3	Rel6	Eth2
8	47	Other	Extended	Nat4	Rel3	Eth3
9	54	Male	Single Man	Nat4	Rel3	Eth6
10	51	Female	Single Woman	Nat5	Rel7	Eth7
11	71	Other	Nuclear	Nat7	Rel7	Eth7
12	62	Male	Single Man	Nat1	Rel4	Eth5
13	30	Other	Nuclear	Nat1	Rel3	Eth2
14	63	Male	Single Man	Nat2	Rel6	Eth4
15	56	Female	Nuclear	Nat2	Rel5	Eth5
16	72	Female	Nuclear	Nat4	Rel1	Eth6
17	41	Other	Extended	Nat1	Rel5	Eth4
18	74	Other	Extended	Nat2	Rel3	Eth6
19	49	Female	Single Woman	Nat2	Rel1	Eth5

Continued on next page

Table D.7 – continued from previous page

ID	Age	Gender	Family	Nationality	Religion	Ethnicity
20	30	Female	Nuclear	Nat2	Rel7	Eth2
21	26	Male	Single Man	Nat2	Rel3	Eth3
22	21	Female	Extended	Nat2	Rel1	Eth7
23	29	Male	Single Man	Nat3	Rel3	Eth1
24	71	Male	Single Father	Nat7	Rel4	Eth3
25	63	Female	Nuclear	Nat3	Rel3	Eth5
26	18	Other	Extended	Nat5	Rel7	Eth2
27	45	Female	Extended	Nat1	Rel7	Eth4
28	36	Male	Extended	Nat5	Rel4	Eth7
29	63	Male	Extended	Nat4	Rel7	Eth6
30	46	Male	Single Father	Nat4	Rel3	Eth1
31	50	Other	Nuclear	Nat4	Rel5	Eth6
32	49	Male	Nuclear	Nat3	Rel6	Eth2
33	49	Male	Nuclear	Nat5	Rel3	Eth7
34	37	Female	Nuclear	Nat5	Rel4	Eth4
35	24	Female	Single Mother	Nat3	Rel3	Eth2
36	65	Female	Extended	Nat5	Rel7	Eth3
37	37	Female	Single Mother	Nat2	Rel7	Eth1
38	58	Other	Extended	Nat2	Rel2	Eth2
39	40	Other	Nuclear	Nat3	Rel5	Eth7
40	34	Other	Single Father	Nat2	Rel6	Eth4
41	36	Male	Extended	Nat7	Rel5	Eth6
42	43	Male	Single Father	Nat5	Rel4	Eth4
43	41	Male	Single Man	Nat3	Rel5	Eth1
44	24	Female	Single Woman	Nat1	Rel3	Eth5
45	25	Female	Extended	Nat2	Rel7	Eth3
46	66	Female	Extended	Nat7	Rel6	Eth7
47	29	Other	Single Woman	Nat2	Rel5	Eth2
48	71	Other	Single Mother	Nat3	Rel7	Eth5
49	47	Male	Nuclear	Nat2	Rel3	Eth1
50	56	Female	Extended	Nat1	Rel2	Eth1
51	34	Female	Single Mother	Nat3	Rel1	Eth3
52	24	Other	Single Father	Nat2	Rel6	Eth5
53	37	Male	Single Father	Nat1	Rel7	Eth4
54	34	Other	Single Father	Nat1	Rel7	Eth4
55	41	Male	Nuclear	Nat6	Rel5	Eth6
56	31	Other	Extended	Nat4	Rel3	Eth5
57	25	Other	Nuclear	Nat4	Rel1	Eth5
58	56	Male	Extended	Nat4	Rel3	Eth6
59	74	Female	Single Mother	Nat5	Rel4	Eth3
60	49	Female	Single Woman	Nat7	Rel1	Eth2
61	56	Male	Single Father	Nat4	Rel5	Eth7
62	59	Other	Extended	Nat1	Rel2	Eth4
63	28	Female	Single Woman	Nat1	Rel2	Eth2
64	20	Male	Single Man	Nat6	Rel7	Eth3
65	66	Female	Nuclear	Nat5	Rel5	Eth3
66	64	Female	Extended	Nat5	Rel3	Eth6
67	59	Male	Single Father	Nat5	Rel3	Eth4

Continued on next page

Table D.7 – continued from previous page

ID	Age	Gender	Family	Nationality	Religion	Ethnicity
68	61	Male	Single Father	Nat5	Rel3	Eth7
69	22	Other	Single Woman	Nat6	Rel1	Eth4
70	35	Male	Extended	Nat1	Rel5	Eth2
71	72	Female	Extended	Nat1	Rel7	Eth2
72	48	Other	Single Father	Nat1	Rel5	Eth6
73	37	Female	Single Mother	Nat5	Rel3	Eth4
74	53	Male	Nuclear	Nat6	Rel5	Eth7
75	54	Male	Extended	Nat3	Rel2	Eth7
76	22	Female	Extended	Nat6	Rel2	Eth1
77	34	Other	Single Woman	Nat3	Rel3	Eth4
78	65	Other	Extended	Nat5	Rel4	Eth3
79	37	Female	Single Mother	Nat2	Rel2	Eth3
80	42	Male	Single Father	Nat1	Rel1	Eth6
81	20	Female	Single Mother	Nat1	Rel1	Eth1
82	40	Male	Single Man	Nat3	Rel2	Eth2
83	34	Male	Single Father	Nat2	Rel1	Eth3
84	19	Other	Extended	Nat1	Rel3	Eth2
85	67	Female	Single Mother	Nat1	Rel1	Eth1
86	43	Male	Single Man	Nat4	Rel4	Eth4
87	33	Female	Single Woman	Nat2	Rel3	Eth3
88	27	Female	Nuclear	Nat3	Rel2	Eth2
89	55	Male	Nuclear	Nat1	Rel4	Eth3
90	23	Other	Single Woman	Nat2	Rel5	Eth4
91	18	Female	Single Woman	Nat2	Rel5	Eth4
92	24	Female	Single Mother	Nat2	Rel5	Eth4
93	29	Male	Extended	Nat5	Rel3	Eth2
94	37	Male	Single Man	Nat1	Rel4	Eth3
95	21	Male	Nuclear	Nat4	Rel1	Eth6
96	69	Female	Single Woman	Nat7	Rel2	Eth1
97	31	Female	Extended	Nat2	Rel5	Eth1
98	26	Other	Single Woman	Nat5	Rel3	Eth5
99	37	Male	Nuclear	Nat1	Rel7	Eth2
100	20	Other	Single Woman	Nat3	Rel2	Eth1

Table D.7: CHC data record 5: Demographic data of TCNs

ID	Age P.	Gender P.	Family P.	Nationality P.	Religion P.	Ethnicity P.	Location P.	Accessibility P.	Rent Period P.	Share-with P.
1	44-50	Female	Single Woman/Single Mother	Mixed	Don't mind	Mixed	L4/L2/L7	No	1/3/2021-1/7/2022	2-3
2	34-43	Don't mind	Nuclear	Don't mind	Don't mind	Don't mind	L1	Yes	1/1/2021-1/7/2022	5
3	44-50	Don't mind	Single Woman/Single Mother	Mixed	Don't mind	Mixed	L3	Don't mind	1/1/2021-1/7/2022	5
4	44-50	Other	Don't mind	Same	Mixed	Same	L4	No	1/1/2021-1/4/2021	2-3
5	34-43/65-120	Female	Extended/Nuclear	Don't mind	Don't mind	Mixed	L2/L3	Yes	3/7/2021-12/2/2022	5
6	65-120	Other	Single Woman/Single Mother	Mixed	Mixed	Same	L3/L6	Yes	1/1/2021-1/7/2021	2-7
7	18-25/26-33/44-50	Male	Extended/Nuclear	Mixed	Don't mind	Don't mind	L4	Don't mind	1/1/2021-1/7/2022	2-7
8	44-50	Don't mind	Single Mother	Same	Mixed	Mixed	L4/L2/L7	Don't mind	1/1/2021-1/7/2022	5
9	Don't mind	Don't mind	Extended	Mixed	Same	Same	L4/L2/L7	No	1/5/2021-1/5/2022	3-5
10	44-50/51-65	Male	Single Man	Don't mind	Same	Mixed	L5	Yes	15/3/2021-1/4/2022	3-5
11	18-25/26-33	Female	Single Woman/Single Mother	Same	Don't mind	Mixed	L1/L2/L3	No	15/3/2021-1/4/2022	1-4
12	51-65	Female	Single Woman/Single Mother	Mixed	Mixed	Don't mind	L4	Yes	1/5/2021-1/5/2022	2-4
13	34-43	Other	Don't mind	Same	Don't mind	Don't mind	L4/L2/L7	Yes	1/5/2021-1/5/2022	2-3
14	18-25/26-33	Male	Single Mother/Single Father/Extended	Don't mind	Don't mind	Don't mind	L2/L3	Don't mind	23/4/2020-17/5/2021	2
15	Don't mind	Female	Single Mother	Don't mind	Don't mind	Same	L3/L6	Don't mind	1/9/2021-1/12/2021	3-5
16	Don't mind	Female	Single Woman/Single Mother	Same	Mixed	Same	L1/L7	No	1/1/2021-1/7/2022	1
17	34-43	Don't mind	Single Mother	Don't mind	Same	Mixed	L3/L5	No	1/1/2021-1/7/2022	2-4
18	51-65/65-120	Male	Single Father/Single Man	Mixed	Same	Mixed	L3/L6	Don't mind	1/3/2021-1/7/2022	3
19	18-25/26-33/34-43	Don't mind	Don't mind	Don't mind	Same	Mixed	L6/L2	Don't mind	1/1/2021-1/7/2021	4
20	26-33	Don't mind	Single Father	Mixed	Mixed	Same	L5	Don't mind	1/1/2021-1/4/2021	5
21	34-43	Female	Single Mother	Same	Same	Don't mind	Don't mind	Don't mind	1/1/2021-1/7/2022	2-7
22	18-25/26-33	Don't mind	Extended	Don't mind	Mixed	Same	L5	No	1/5/2021-1/5/2022	2-7
23	34-43	Female	Single Mother/Single Father/Extended	Mixed	Don't mind	Don't mind	L4/L2/L7	Yes	1/1/2021-1/7/2022	1
24	Don't mind	Female	Single Mother	Same	Don't mind	Mixed	L1	Don't mind	15/3/2021-1/4/2022	4
25	Don't mind	Female	Single Woman/Single Mother	Same	Don't mind	Same	L6	No	1/1/2021-1/7/2021	4
26	Don't mind	Don't mind	Don't mind	Same	Don't mind	Mixed	L3	Don't mind	Don't mind	2-7
27	34-43/51-65	Don't mind	Nuclear	Mixed	Same	Mixed	L3	Yes	1/9/2021-1/12/2021	2-7
28	18-25/26-33	Female	Single Mother	Mixed	Mixed	Same	L4/L2/L7	No	1/1/2021-1/7/2021	3-5
29	Don't mind	Other	Don't mind	Same	Don't mind	Mixed	L4/L2/L7	Yes	1/3/2021-1/7/2022	1-2
30	44-50	Don't mind	Single Father	Mixed	Same	Don't mind	L7	Yes	1/9/2021-1/12/2021	2-3
31	65-120	Male	Nuclear	Don't mind	Same	Don't mind	Don't mind	Don't mind	Don't mind	1
32	34-43	Male	Single Father	Don't mind	Don't mind	Don't mind	L6	Don't mind	1/1/2021-1/7/2022	Don't mind
33	Don't mind	Other	Extended	Mixed	Same	Mixed	L3/L6/L7	No	1/1/2021-1/7/2022	1-5
34	34-43	Don't mind	Extended/Nuclear	Same	Mixed	Mixed	L7	Don't mind	Don't mind	1-5
35	18-25	Male	Extended/Nuclear	Don't mind	Mixed	Same	L1/L7	No	15/3/2021-1/4/2022	1-2
36	44-50/51-65/65-120	Female	Extended	Mixed	Same	Same	L6/L2	Yes	1/1/2021-1/7/2021	3
37	Don't mind	Don't mind	Don't mind	Don't mind	Don't mind	Mixed	L3/L6	Yes	1/9/2021-1/12/2021	3
38	65-120	Female	Single Woman	Mixed	Mixed	Mixed	L5	Don't mind	1/9/2021-1/12/2021	1-4
39	18-25/26-33	Other	Don't mind	Don't mind	Don't mind	Mixed	Don't mind	No	1/9/2021-1/12/2021	1-3
40	44-50/51-65	Don't mind	Single Father	Don't mind	Don't mind	Mixed	L7/L6/L5	Don't mind	1/5/2021-1/5/2022	1-2
41	44-50/51-65	Don't mind	Extended	Same	Mixed	Same	L2	Yes	15/3/2021-1/4/2022	3
42	18-25/26-33/34-43	Female	Single Woman/Single Mother	Same	Same	Mixed	L2	No	3/7/2021-12/2/2022	1
43	34-43	Don't mind	Single Man	Don't mind	Don't mind	Same	L2/L3	Don't mind	1/1/2021-1/7/2021	3-4
44	18-25	Female	Don't mind	Mixed	Same	Mixed	L1	Yes	1/1/2021-1/7/2021	1-5
45	18-25/26-33	Female	Nuclear	Same	Don't mind	Mixed	L1/L2/L3	Yes	1/5/2021-1/5/2022	1
46	51-65/65-120	Male	Don't mind	Same	Mixed	Mixed	L5	Don't mind	1/1/2021-1/7/2022	5
47	18-25/26-33/34-43	Male	Single Father/Single Man	Don't mind	Same	Mixed	Don't mind	Don't mind	23/4/2020-17/5/2021	1-2
48	65-120	Other	Single Father/Single Man	Mixed	Don't mind	Same	L6/L2	No	1/5/2021-1/5/2022	3-5
49	Don't mind	Female	Nuclear	Mixed	Same	Mixed	L5	Yes	1/1/2021-1/4/2021	1-5
50	51-65/65-120	Female	Single Woman/Single Mother	Don't mind	Don't mind	Mixed	L3/L6 Don't mind	1/1/2021-1/4/2021	2-7	
51	26-33/34-43	Don't mind	Don't mind	Mixed	Mixed	Mixed	L4	Yes	3/7/2021-12/2/2022	2-4
52	44-50/51-65	Don't mind	Single Mother/Single Father/Extended	Don't mind	Mixed	Mixed	L4	Don't mind	1/1/2021-1/7/2022	3-4
53	18-25	Male	Single Man	Same	Same	Don't mind	L4	Yes	1/5/2021-1/5/2022	Don't mind
54	51-65/65-120	Other	Extended	Mixed	Mixed	Don't mind	L3/L6	Yes	1/1/2021-1/7/2021	3
55	51-65	Female	Extended	Same	Mixed	Don't mind	L3/L5	Yes	1/1/2021-1/7/2022	5
56	51-65/65-120	Other	Extended/Nuclear	Same	Same	Don't mind	L7/L6/L5	No	Don't mind	1-3
57	Don't mind	Other	Single Woman/Single Mother	Mixed	Mixed	Mixed	L3	No	1/3/2021-1/7/2022	2-3
58	Don't mind	Male	Single Mother/Single Father/Extended	Mixed	Same	Mixed	L4/L2/L7	Don't mind	15/3/2021-1/4/2022	1-5
59	44-50/51-65	Female	Single Woman/Single Mother	Same	Same	Don't mind	Don't mind	Yes	1/3/2021-1/7/2022	2-7
60	18-25/26-33/34-43	Female	Single Woman/Single Mother	Don't mind	Mixed	Don't mind	L1	Don't mind	1/1/2021-1/4/2021	3-5
61	44-50/51-65	Don't mind	Don't mind	Mixed	Don't mind	Same	L1	Yes	1/3/2021-1/7/2022	2-3
62	65-120	Female	Single Mother/Single Father/Extended	Mixed	Mixed	Mixed	L2/L3 Yes	Don't mind	2-7	
63	34-43	Other	Extended/Nuclear	Don't mind	Same	Mixed	L4/L2/L7	Don't mind	3/7/2021-12/2/2022	5
64	18-25	Male	Don't mind	Same	Same	Same	L4	No	1/1/2021-1/7/2021	3-5
65	34-43	Female	Single Woman/Single Mother	Mixed	Don't mind	Same	L1	Don't mind	1/1/2021-1/4/2021	1-3
66	Don't mind	Female	Single Mother/Single Father/Extended	Mixed	Same	Mixed	L2/L3	Yes	1/1/2021-1/7/2021	2-4
67	18-25/26-33/34-43	Male	Don't mind	Don't mind	Same	Same	L7/L6/L5	Don't mind	1/1/2021-1/4/2021	2-7
68	44-50	Don't mind	Single Man	Mixed	Same	Same	L6	Don't mind	1/1/2021-1/7/2021	1-2
69	Don't mind	Don't mind	Don't mind	Don't mind	Same	Mixed	L5	No	1/3/2021-1/7/2022	1-5
70	44-50/51-65	Don't mind	Don't mind	Don't mind	Mixed	Same	L3/L5	Don't mind	1/1/2021-1/7/2021	1-2
71	51-65/65-120	Female	Don't mind	Don't mind	Same	Don't mind	L5	Yes	23/4/2020-17/5/2021	Don't mind
72	51-65	Don't mind	Extended/Nuclear	Mixed	Don't mind	Don't mind	L4	No	1/1/2021-1/7/2022	2-4
73	44-50	Female	Nuclear	Mixed	Same	Don't mind	L6/L2	Don't mind	Don't mind	1-3
74	51-65/65-120	Male	Extended	Same	Mixed	Mixed	L4/L2/L7	Yes	15/3/2021-1/4/2022	2
75	18-25/26-33/34-43	Male	Extended/Nuclear	Same	Don't mind	Don't mind	L7/L6/L5	Yes 1/5/2021-1/5/2022	Don't mind	
76	18-25/26-33/34-43	Female	Single Mother/Single Father/Extended	Mixed	Don't mind	Don't mind	L6	Yes	1/1/2021-1/7/2022	1-5
77	26-33/34-43/65-120	Male	Single Man	Mixed	Don't mind	Same	L2/L3	Yes	23/4/2020-17/5/2021	1
78	65-120	Male	Don't mind	Same	Mixed	Mixed	L7	Yes	1/1/2021-1/7/2021	1-2
79	51-65	Male	Single Mother/Single Father/Extended	Don't mind	Mixed	Same	L7/L6/L5	Don't mind	1/5/2021-1/5/2022	Don't mind
80	51-65	Male	Single Man	Don't mind	Mixed	Mixed	L1/L2/L3	No	1/3/2021-1/7/2022	5
81	18-25	Female	Single Woman/Single Mother	don't mind	don't mind	Mixed	don't mind	don't mind	1/1/2021-1/7/2021	2-3
82	don't mind	Mixed	don't mind	Mixed	don't mind	Same	L1/L2/L3	don't mind	1/5/2021-1/5/2022	Don't mind
83	26-33/34-43	Don't mind	Single Mother/Single Father/Nuclear	don't mind	Same	Don't mind	L3/L4/L5	Yes	1/1/2021-1/7/2022	2-4
84	18-25	Don't mind	Don't mind	don't mind	Same	Mixed	don't mind	don't mind	Don't mind Don't mind	
85	Don't mind	Female	Single Woman/Single Mother/Nuclear	Same	Same	Don't mind	don't mind	Yes	Don't mind	1-2
86	34-43/44-50	Don't mind	Don't mind	don't mind	don't mind	Mixed	don't mind	don't mind	1/3/2021-1/7/2022	1-3
87	18-25/26-33/34-43	Female	Single Woman/Single Mother	Same	don't mind	Mixed	L1/L2/L3/L4	No	1/1/2021-1/7/2022	Don't mind
88	Don't mind	Don't mind	Single Woman/Single Mother/Nuclear	Mixed	Mixed	Same	L3/L6/L9	Yes	Don't mind	1-4
89	51-65/65-120	Male	Don't mind	Same	Same	Same	L4	Yes	1/5/2021-1/12/2021	1-2
90	Don't mind	Female	Single Woman/Single Mother	don't mind	don't mind	Don't mind	don't mind	No	1/1/2021-1/12/2021	Don't mind
91	18-25	Female	Single Woman/Single Mother	Same	Same	Same	L7/L8	No	1/1/2021-1/4/2021	3
92	Don't mind	Don't mind	Single Woman/Single Mother/Nuclear	Same	Same	Same	L3/L6/L7/L9	Don't mind	Don't mind	Don't mind
93	26-33/34-43	Don't mind	Extended	Don't mind	Same	Don't mind	L1/L2/L3/L4	Yes	1/9/2021-1/2/2022	Don't mind
94	Don't mind	Don't mind	Single Man/Single Father	Mixed	don't mind	Don't mind	don't mind	Don't mind	Don't mind	1-3
95	18-25	Mixed	Don't mind	Same	Mixed	Mixed	L3/L12	don't mind	1/4/2021-1/4/2022	2-3
96	51-65/65-120	Female	Don't mind	Same	Same	Same	Don't mind	Yes	1/7/2021-1/12/2021	Don't mind
97	18-25/26-33/34-43	Female	Single Woman/Single Mother/Extended	Don't mind	don't mind	don't mind	L4/L7/L12/L14	Don't mind	1/1/2021-1/7/2022	1-2
98	18-25/26-33	Other	Don't mind	Same	Don't mind	Mixed	don't mind	Don't mind	Don't mind	Don't mind
99	26-33/34-43	Don't mind	Nuclear	Same	Mixed	Same	L2/L3/L4	Yes	15/3/2022-15/10/2022	Don't mind
100	18-25	Female	Single Mother	Mixed	Mixed	Same	L1/L11	Yes	5/1/2021-5/7/2021	2-3

Table D.8: CHC data record 5: Individual Preferences of TCNs

D.2 Results

This section presents the solutions that were computed for the CHC data records (section D.1) and configurations (tables 5.1, 5.4, and 5.5).

1. Result of the pair of **Configuration 1** (table 5.1) and data record 1 (table 5.3) is presented in section 5.4.
2. Result of the pair of **Configuration 2** (table 5.4) and data record 1 (table 5.3) is presented in section 5.4.
3. Result of the pair of **Configuration 3** (table 5.5) and data record 1 (table 5.3) is presented in section 5.4.
4. Result of the pair of **Configuration 1** (table 5.1) and data record 2 (table D.2) is presented below.
   - **Total Computation time**: 60 seconds
   - **Solution**:
     Singletons: [2, 5, 6, 9, 10, 20]
     Group 1: [1, 16]
     Group 2: [15, 17]
     Group 3: [3, 7, 8, 12, 13, 19, 4, 11, 14, 18]
   - **Customer Satisfaction Score**: 4
5. Result of the pair of **Configuration 2** (table 5.4) and data record 2 (table D.2) is presented below.
   - **Total Computation time**: 61 seconds
   - **Solution**:
     Singletons: [5]
     Group 1: [2, 6, 9, 10, 20]
     Group 2: [1, 16]
     Group 3: [4, 7, 8, 11, 12, 13, 14, 19]
     Group 4: [3, 15, 17, 18]
   - **Customer Satisfaction Score**: 4.2
6. Result of the pair of **Configuration 3** (table 5.5) and data record 2 (table D.2) is presented below.
   - **Total Computation time**: 64 seconds
   - **Solution**:
     Singletons: [13, 14]
     Group 1: [2, 5, 6, 8, 9, 15, 17, 20]
     Group 2: [1, 16]
     Group 3: [4, 10]
     Group 4: [7, 11, 12, 19]
     Group 5: [3, 18]
   - **Customer Satisfaction Score**: 4.28

7. Result of the pair of **Configuration 1** (table 5.1) and data record 3 (table D.4) is presented below.
   - **Total Computation time**: 67 seconds
   - **Solution**:
     Singletons: [3, 5, 6, 7, 9, 10, 13, 19, 1, 11, 14, 18, 20, 21, 23]
     Group 1: [15, 24]
     Group 2: [2, 8, 17, 16, 25]
     Group 3: [12, 22, 4]
   - **Customer Satisfaction Score**: 4.25
8. Result of the pair of **Configuration 2** (table 5.4) and data record 3 (table D.4) is presented below.
   - **Total Computation time**: 72 seconds
   - **Solution**:
     Group 1: [1, 3, 6, 7, 11, 13, 18, 20]
     Group 2: [14, 23]
     Group 3: [5, 9, 10, 12, 15, 17, 19]
     Group 4: [2, 8, 16, 25]
     Group 5: [4, 22]
   - **Customer Satisfaction Score**: 4
9. Result of the pair of **Configuration 3** (table 5.5) and data record 3 (table D.4) is presented below.
   - **Total Computation time**: 65 seconds
   - **Solution**:
     Group 1: [1, 7, 18]
     Group 2: [11, 20]
     Group 3: [2, 3, 4, 5, 6, 8, 9, 10, 12, 13, 14, 15, 16, 17, 19, 21, 22, 23, 24, 25]
   - **Customer Satisfaction Score**: 4.5
10. Result of the pair of **Configuration 1** (table 5.1) and data record 4 (table D.6) is presented below.
    - **Total Computation time**: 225 seconds
    - **Solution**:
      Singletons: [2, 3, 5, 7, 8, 10, 12, 13, 17, 19, 29, 35, 42, 44, 47, 1, 4, 14, 18, 20, 21, 23, 36, 40, 41, 43, 45, 50]
      Group 1: [6, 11, 25, 46]
      Group 2: [9, 24, 38]
      Group 3: [28, 39]
      Group 4: [15, 49]
      Group 5: [27, 37, 16]
      Group 6: [30, 32, 31]
      Group 7: [26, 34]

Group 8: [22, 33, 48]

- **Customer Satisfaction Score**: 4.2

11. Result of the pair of **Configuration 2** (table 5.4) and data record 4 (table D.6) is presented below.
    - **Total Computation time**: 214 seconds
    - **Solution**:
      Singletons: [21]
      Group 1: [2, 3, 4, 7, 8, 13]
      Group 2: [6, 19, 25, 43]
      Group 3: [18, 47]
      Group 4: [1, 12]
      Group 5: [5, 42]
      Group 6: [11, 46]
      Group 7: [10, 20, 24, 44]
      Group 8: [9, 17, 22, 23, 38, 50]
      Group 9: [15, 28, 29, 39, 48]
      Group 10: [16, 27, 37]
      Group 11: [30, 31, 32]
      Group 12: [26, 34]
      Group 13: [33, 48]
    - **Customer Satisfaction Score**: 4.64
12. Result of the pair of **Configuration 3** (table 5.5) and data record 4 (table D.6) is presented below.
    - **Total Computation time**: 231 seconds
    - **Solution**:
      Singletons: [15]
      Group 1: [2, 3, 4, 5, 7, 8, 11, 13, 14, 18, 20, 35, 36, 40, 41, 42, 43, 44, 45, 47]
      Group 2: [19, 23]
      Group 3: [9, 12, 25]
      Group 4: [16, 21]
      Group 5: [1, 17, 46]
      Group 6: [6, 24, 38]
      Group 7: [10, 22, 33, 48]
      Group 8: [26, 34, 50]
      Group 9: [27, 37]
      Group 10: [30, 31, 32]
      Group 11: [28, 39]
      Group 12: [29, 49]
    - **Customer Satisfaction Score**: 4.48
13. Result of the pair of **Configuration 1** (table 5.1) and data record 5 (table D.8) is presented below.
    - **Total Computation time**: 514 seconds

- **Solution**:
  Singletons: [1, 3, 4, 5, 6, 7, 8, 11, 13, 14, 15, 17, 19, 20, 21, 22, 23, 25, 27, 28, 30, 33, 34, 35, 38, 41, 42, 43, 44, 45, 47, 48, 49, 50, 52, 53, 54, 55, 56, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 72, 73, 74, 75, 77, 78, 79, 80, 81, 82, 83, 84, 85, 89, 95, 96, 100]
  Group 1: [16, 31]
  Group 2: [24, 46]
  Group 3: [10, 12]
  Group 4: [9, 18]
  Group 5: [36, 71]
  Group 6: [29, 40]
  Group 7: [58, 93]
  Group 8: [26, 98]
  Group 9: [2, 88, 99]
  Group 10: [39, 57]
  Group 11: [32, 37, 51, 76, 86, 87, 94, 97]
  Group 12: [90, 91, 92]

- **Customer Satisfaction Score**: 4.2

14. Result of the pair of **Configuration 2** (table 5.4) and data record 5 (table D.8) is presented below.

    - **Total Computation time**: 498 seconds

    - **Solution**:
      Singletons: [14, 62]
      Group 1: [4, 6, 8, 11, 16, 25, 30, 31, 35, 38, 41, 42, 48, 53, 54, 55, 56, 59, 63, 64, 67, 68, 74, 78, 89, 95, 96, 100]
      Group 2: [24, 46]
      Group 3: [7, 23, 28, 61, 75, 79, 85]
      Group 4: [49, 73]
      Group 5: [9, 18, 72]
      Group 6: [10, 12]
      Group 7: [22, 34, 52, 65, 80, 82]
      Group 8: [29, 40]
      Group 9: [1, 44]
      Group 10: [36, 71]
      Group 11: [58, 93]
      Group 12: [26, 98]
      Group 13: [13, 15, 17, 21, 33, 43, 47, 50, 66, 77, 81, 83, 84]
      Group 14: [39, 57]
      Group 15: [19, 69]
      Group 16: [5, 20]
      Group 17: [2, 99]
      Group 18: [3, 27, 32, 45, 60, 70, 76, 87, 88]
      Group 19: [37, 51, 86, 94, 97]
      Group 20: [90, 91, 92]

    - **Customer Satisfaction Score**: 4.77

15. Result of the pair of **Configuration 3** (table 5.5) and data record 5 (table D.8) is

presented below.

- **Total Computation time**: 517 seconds
- **Solution**:
  Group 1: [11, 16, 20, 24, 25, 41, 42, 48, 59, 63, 64, 74]
  Group 2: [46, 55]
  Group 3: [6, 8, 28, 38, 54, 68, 96]
  Group 4: [36, 71]
  Group 5: [4, 5, 10, 17, 18, 21, 23, 31, 35, 44, 45, 49, 50, 53, 56, 65, 67, 73, 75, 78, 89, 95, 100]
  Group 6: [1, 7, 34, 80]
  Group 7: [2, 82]
  Group 8: [79, 83]
  Group 9: [3, 14, 22, 29, 30, 40, 47, 61, 69, 72, 77, 85]
  Group 10: [13, 19, 27, 60, 70, 76, 81]
  Group 11: [52, 66]
  Group 12: [39, 57]
  Group 13: [9, 33, 84]
  Group 14: [58, 93]
  Group 15: [26, 98]
  Group 16: [12, 43]
  Group 17: [90, 91, 92]
  Group 18: [88, 99]
  Group 19: [15, 37, 62]
  Group 20: [32, 51, 86, 94]
  Group 21: [87, 97]

- **Customer Satisfaction Score**: 4.7