EGE ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİGİ BÖLÜMÜ

ÇOK ETMENLİ YAPAY ZEKA

YÜKSEK LISANS DERSI

2023 – 2024 GÜZ DÖNEMİ

PROJE ÖDEVİ

Dr. BİROL ÇİLOĞLUGİL

MOHAMMAD MAHDI SARHANGI

91230000255

Literature Overview

Agent-Oriented Programming (AOP):

AOP explores the concept of using agents as the primary unit of computation in software design. Agents are autonomous entities capable of perceiving their environment, reasoning about it, and acting accordingly, leading to more flexible and adaptable software systems. This approach is believed to better manage the complexity and uncertainty of real-world environments, potentially leading to smarter and more responsive systems.

Shoham, Y. (1993). "Agent-oriented programming." Artificial Intelligence, 60, 51-92.

Multi-Agent Systems (MAS):

MAS involves the creation of software systems composed of multiple interacting agents. These agents can be autonomous or semi-autonomous, and they can exist in physical or virtual forms. The goal of MAS is to create systems that can adapt to changing environments, coordinate their actions, and achieve a common objective.

Jain, L., & Wellman, M. (2001). "Multi-agent systems." Communications of the ACM, 44(8), 71-82.

Approaches in Multi-Agent Programming:

Various approaches include rule-based systems (Rao & Georgeff, 1995), reactive systems (Wooldridge, 2002), and negotiation systems (Nguyen & Durfee, 2002). Reactive systems are designed to respond quickly to environmental changes, while rule-based systems depend on predefined rules to govern agent behavior. Negotiation systems are designed for considering multiple options and making decisions based on a set of objectives.

Rao, A., & Georgeff, M. (1995). "Modeling rational agents within a BDI-architecture." Journal of Logic and Computation, 5(4), 499-550.
Wooldridge, M. (2002). An introduction to multi-agent systems. John Wiley & Sons.
JaCaMo Framework:

JaCaMo is an integration of three platforms:

Jason for programming agents, Moise for organizational aspects, and CArtAgO for shared environments. It brings together agent-oriented programming, organization-oriented programming, and environment-oriented programming, realizing the full potential of MAS as a programming paradigm. JaCaMo is used for creating systems where coordination, communication, and negotiation among agents are crucial.

Boissier, O., Bordini, R. H., Hübner, J. F., Ricci, A., & Santi, A. (2013). "Multi-agent oriented programming with JaCaMo." Science of Computer Programming, 78, 747–761.

Recent Trends in Multi-Agent Programming:

Recent research in multi-agent programming has focused on aspects such as coordination (Kraus, 2000), communication (Jenkins, 2002), and negotiation (Rosenschein & Zlotkin, 1994). These research areas have been applied to various fields including distributed systems, robotics, and intelligent environments.

Kraus, S. (2000). "Coordination in multi-agent systems." Journal of Logic and Computation, 10(5), 637-653.

Application Areas:

JaCaMo and MAS have been applied in diverse domains such as distributed systems (Sycara, 1998), robotics (Simmons, 1997), and intelligent environments, demonstrating their versatility and efficacy in handling complex, distributed, and dynamic systems.

Simmons, R. (1997). "Multi-agent systems and robotics." AI Magazine, 18(3), 57-68.

Project Overview

This project involves the development of a multi-agent system (MAS) using JaCaMo, designed for a hospital environment focusing on complex challenges within the healthcare domain through the development of a Multi-Agent System (MAS) using the JaCaMo platform. Integration of advanced technologies and intelligent systems has become critical in ensuring efficient, patient-centric care in the last years.

The Multi-Agent System (MAS) for healthcare management consists of system components designed to collaboratively address diverse healthcare scenarios. At its core are intelligent agents representing distinct roles, including **doctors, paramedics, nurses, and patients**, each equipped with specialized functionalities and decision-making capabilities.

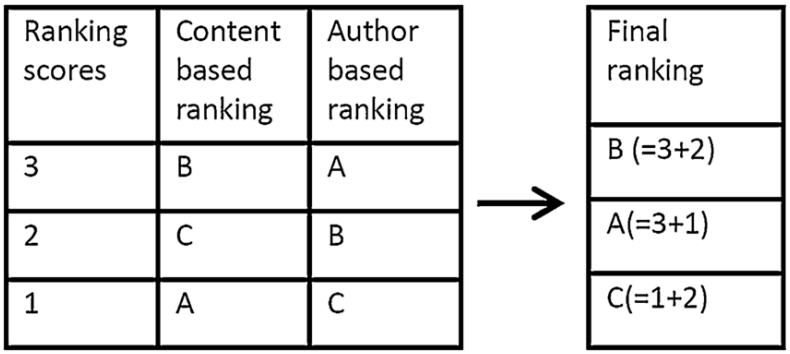
These agents interact with artifacts such as the **HeartRateMonitor** and **PatientCareSystem**, which centralize critical functionalities such as heart rate monitoring, decision-making processes, and patient prioritization.

The organizational specifications, **emergency_room_org.xml** and **healthcare_org.xml**, delineate the roles, group structures, functional goals, and normative rules, fostering a structured and rule-based environment. These organizational guidelines ensure that agents operate cohesively within their designated roles, contributing to decentralized decision-making and effective patient care.

Borda Count Algorithm:

“The Borda count is a family of positional voting rules which gives each candidate, for each ballot, a number of points corresponding to the number of candidates ranked lower. In the original variant, the lowest-ranked candidate gets 0 points, the next-lowest gets 1 point, etc., and the highest-ranked candidate gets n − 1 points, where n is the number of candidates. Once all votes have been counted, the option or candidate with the most points is the winner.”

In our project we used the Borda count algorithm in the **PatientCareSystem** artifact in order to choose the highest priority patient in the emergency room and give the chosen patient treatment by the nurses available in the system. **Nurses** will choose their priorities and send it to the doctor agent, **doctor** agent will use **PatientCareSystem** artifact which utilizes the Borda count algorithm and decides which patient has the highest priority.



Ranking scores	Content based ranking	Author based ranking
3	B	A
2	C	B
1	A	C

Final ranking
B (=3+2)
A(=3+1)
C(=1+2)

System Components:

Agents:

Doctor Agent:

The Doctor Agent, focuses on patient care and heart rate monitoring. Using the Borda count algorithm, the Doctor Agent participates in decision-making processes, contributing to the prioritization of patients. This agent's responsibilities include monitoring and responding to patient conditions, ensuring effective communication within the healthcare team, and making informed decisions based on the overall well-being of patients.

```
/* Initial beliefs and rules */

chosen(W) :- ballots(B) & borda(B,R) & .max(R,count(_,W)).

borda(B,R) :- patients(C) & .length(C,N) & borda_count(B,[],N,R).

borda_count([],R,_,R).
borda_count([V|T],I,N,R) :- count_vote(V,I,1,N,J) & borda_count(T,J,N,R).

count_vote([],R,_,_,R).
count_vote([C|T],L,X,Y,R) :- .member(count(Z,C),L) & .delete(count(Z,C),L,L1) & count_vote(T,[count(Z+(Y-X),C)|L1],X+1,Y,R).
count_vote([C|T],L,X,Y,R) :- count_vote(T,[count((Y-X),C)|L],X+1,Y,R).



/* Initial goals */

+!open <- open([patient1,patient2,patient3],[nurse1,nurse2]).
+!close <- .wait(1000); close; !declare_chosen.

+!declare_chosen: chosen(W) <- .print("The chosen patient is: ",W); .send(W,tell,chosen(healthcare_team)).


{ include("$jacamoJar/templates/common-cartago.asl") }
{ include("$jacamoJar/templates/common-moise.asl") }
{ include("$jacamoJar/templates/org-obedient.asl") }

```

Paramedic Agent:

The Paramedic Agent is specialized in monitoring heart rates and initiating actions based on predefined thresholds. This agent is responsible for ensuring the safety and well-being of patients by actively monitoring heart rates. In instances where a patient's heart rate exceeds safe thresholds, the Paramedic Agent takes appropriate actions, such as alerting the healthcare team or initiating emergency protocols. The Paramedic Agent contributes to the overall patient care strategy by providing real-time information on critical health parameters.

```
now_is_unsafe_rate(R) :- heart_rate(H) & H > R.

+!heart_rate(R):
    now_is_unsafe_rate(R) &
    heart_rate(H)
    <-
    if (not state("monitoring")) {
        startMonitoring;
        .log(warning, H, " is outside safe range -> monitoring until ", R);
    }
    !heart_rate(R);
.

+!heart_rate(R):
    state("monitoring")
    <-
    stopMonitoring;
    .log(warning,"Heart rate is now within the safe range.");
    !heart_rate(R);
.

+!heart_rate(R)
    <-
    !heart_rate(R);
.

+!add_preference(R)[source(S)]
    <-
    .abolish(preference(S,_));
    +preference(S,R);
    .findall(X,preference(_,X),L);
    .drop_desire(heart_rate(_));
    !heart_rate(math.mean(L));
.

{ include("$jacamoJar/templates/common-cartago.asl") }
{ include("$jacamoJar/templates/common-moise.asl") }
```

Nurse Agents:

Nurse Agents are actively engaged in patient care and treatment processes. These agents play an important role in the voting mechanism for patient prioritization, leveraging their insights and expertise to contribute to decision-making. Nurse Agents also facilitate communication within the healthcare team, ensuring that information flows seamlessly between different entities. Their responsibilities include adhering to normative specifications, participating in the treatment of patients, and collaborating with other agents to provide comprehensive healthcare services.

```
+!vote_patient : my_priorities(V) <- vote(V).

+!treatment_process : medical_condition(X) <- .print("Helping on condition ",X); !send_preference.

+!send_preference(R)
    <-
    -+recipient_agent(R);
    !send_preference;
.

+!send_preference:
    preferred_heart_rate_range(H) &
    recipient_agent(R)
    <-
    .log(warning,"Sending preference for heart rate range ", H);
    .send(R,achieve,add_preference(H));
.

{ include("$jacamoJar/templates/common-cartago.asl") }
{ include("$jacamoJar/templates/common-moise.asl") }
{ include("$jacamoJar/templates/org-obedient.asl") }
```

Patient Agents:

Patient Agents represent individuals within the MAS who undergo treatment based on their medical conditions. These agents actively participate in the decision-making process by contributing to the voting mechanism for patient prioritization. Patient Agents communicate their medical conditions to the healthcare team, allowing for personalized and targeted care. Additionally, they play a role in the overall patient care strategy, providing valuable input through the voting process to influence the prioritization of treatments and interventions.

```
1
2   +chosen(healthcare_team) : .my_name(Me) & play(Me,normal,Org) <-
3       .findall(C,play(C,normal,Org),CS);
4       ?medical_condition(T);
5       .send(CS, tell, medical_condition(T));
6       adoptRole(pathological)[artifact_name(Org)].
7
8   +!tell_condition : medical_condition(X) <- .print("I'm having the condition ",X).
9
10  +!treatment_process : medical_condition(X) <- .print("I'm getting treatment on condition ",X); !set_heart_rate.
11
12  +!set_heart_rate: heart_rate(H) <- .print("My heart rate is ",H); setHeartRate(H);.
13
14  { include("$jacamoJar/templates/common-cartago.asl") }
15  { include("$jacamoJar/templates/common-moise.asl") }
16  { include("$jacamoJar/templates/org-obedient.asl") }
17
```

Artifacts:

PatientCareSystem Artifact:

The PatientCareSystem artifact serves as the orchestrator of the patient prioritization process within the healthcare system. This artifact manages the voting mechanism, allowing doctors, nurses, and patients to contribute to the decision-making process. By overseeing the opening, voting, and closing phases, it facilitates efficient communication among agents and ensures a decentralized approach to patient prioritization. The artifact's modular design enhances the adaptability of the system, making it a central component in the collaborative decision-making framework of the MAS.

```java
package tools;

import java.util.List;
import java.util.ArrayList;

import cartago.*;
import jason.asSyntax.ASSyntax;
import jason.asSyntax.ListTerm;
import jason.asSyntax.parser.ParseException;

public class PatientCareSystem extends Artifact {

    List<String> voters;
    List<Object[]> result = new ArrayList<Object[]>();

    public void init() {
        defineObsProperty("status", "closed");
    }

    @OPERATION public void open(Object[] patients, Object[] voters) {
        this.voters  = new ArrayList<String>();

        ListTerm cs = ASSyntax.createList();
        for (Object o: patients)
            try {
                cs.add(ASSyntax.parseTerm(o.toString()));
            } catch (ParseException e) {
                e.printStackTrace();
            }
        for (Object o: voters)
            this.voters.add(o.toString());

        defineObsProperty("patients", cs);
        getObsProperty("status").updateValue("open");

    }

    @OPERATION void vote(Object[] vote) {
        if (getObsProperty("status").getValue().equals(obj:"close")) {
            failed("the voting machine is closed!");
        }
        if (voters.remove(getCurrentOpAgentId().getAgentName())) {
                result.add(vote);
        } else {
            failed("you voted already!");
        }
    }

    @OPERATION void close() {
        ListTerm final_result = ASSyntax.createList();
        for (Object[] i: result) {
            ListTerm l = ASSyntax.createList();
            for (Object j : i) {
                try {
                    l.add(ASSyntax.parseTerm(j.toString()));
                } catch (ParseException e) {
                    e.printStackTrace();
                }
            }
            final_result.add(l);
        }
        getObsProperty("status").updateValue("closed");
        defineObsProperty("ballots", final_result);
    }

}

```

HeartRateMonitor Artifact:

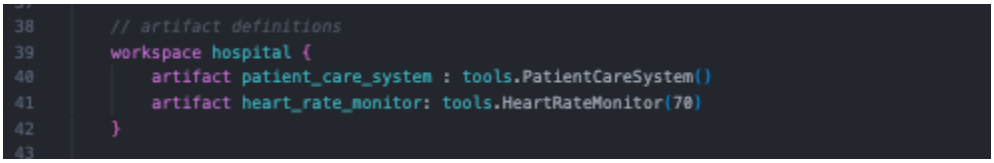
The HeartRateMonitor artifact plays a crucial role in simulating and monitoring the heart rates of patients within the healthcare system. Through its methods, such as **setHeartRate** and **startMonitoring**, it enables paramedics to monitor and respond to changes in patients' heart rates. The artifact, with its internal operation **updateHeartRateProc**, emulates continuous heart rate updates during monitoring. By providing a standardized interface for heart rate management, this artifact contributes to the system's ability to respond dynamically to health-related events and supports effective patient care.

```
package tools;

import cartago.*;

@ARTIFACT_INFO(outports = { @OUTPORT(name = "out-1") })
public class HeartRateMonitor extends Artifact {

    void init(double initialHeartRate){
        defineObsProperty("state","idle");
        defineObsProperty("heart_rate", initialHeartRate);
        log("Heart Rate: "+ getObsProperty("heart_rate").doubleValue());
    }

    @OPERATION void setHeartRate(double heartRate){
        getObsProperty("heart_rate").updateValue(heartRate);
        log("Heart Rate: "+ getObsProperty("heart_rate").doubleValue());
    }

    @OPERATION void startMonitoring(){
        log("startMonitoring");
        getObsProperty("state").updateValue("monitoring");
        this.execInternalOp("updateHeartRateProc", -1.0); // Simulates heart rate changes
    }

    @OPERATION void stopMonitoring(){
        log("stopMonitoring");
        getObsProperty("state").updateValue("idle");
    }

    @INTERNAL_OPERATION void updateHeartRateProc(double step){
        ObsProperty heartRate = getObsProperty("heart_rate");
        ObsProperty state = getObsProperty("state");
        while (!state.stringValue().equals(anObject:"idle")){
            heartRate.updateValue(heartRate.doubleValue() + step);
            log("Heart Rate: "+ heartRate.doubleValue());
            this.await_time(1000); // Adjust time interval as needed
        }
    }
}
```

Workspace:

hospital:

The hospital workspace within the Multi-Agent System (MAS) serves as the central environment where healthcare activities unfold. Defined as the "hospital" artifact in the MAS project, this workspace encapsulates crucial artifacts such as the PatientCareSystem and HeartRateMonitor. The PatientCareSystem artifact manages the intricate process of patient prioritization through a voting mechanism, engaging doctors, nurses, and patients in decentralized decision-making. Simultaneously, the HeartRateMonitor artifact simulates and monitors heart rates, providing essential data for timely interventions by paramedics.



Organization:

Emergency Room Organization (emergency_room_org.xml):

The Emergency Room Organization (emergency_room_org.xml) establishes a structured framework for role-based coordination within an emergency medical scenario.

The organizational specification defines two roles, **"pathological"** and **"normal"**, representing agents in critical and stable conditions, respectively.

The group specification, denoted as emergency_room_grp, delineates the allowable roles and their relationships within the emergency room setting. The roles are interconnected with links, with the "pathological" role possessing an authority over the "normal" role in intra-group interactions.

The formation constraints ensure compatibility between these roles. The functional specification, encapsulated in the **emergency_room_sch scheme**, outlines missions and goals to be achieved by the agents. Notably, the **"treatment"** goal orchestrates a sequence of sub-goals, including informing about medical conditions and undergoing treatment.

Normative specifications define obligations, such as ensuring that agents with the **"pathological"** role fulfill the **"telling_mission"**. In summary, this organizational structure

orchestrates a dynamic and well-defined emergency room environment, where roles, goals, and norms collectively contribute to effective healthcare management.

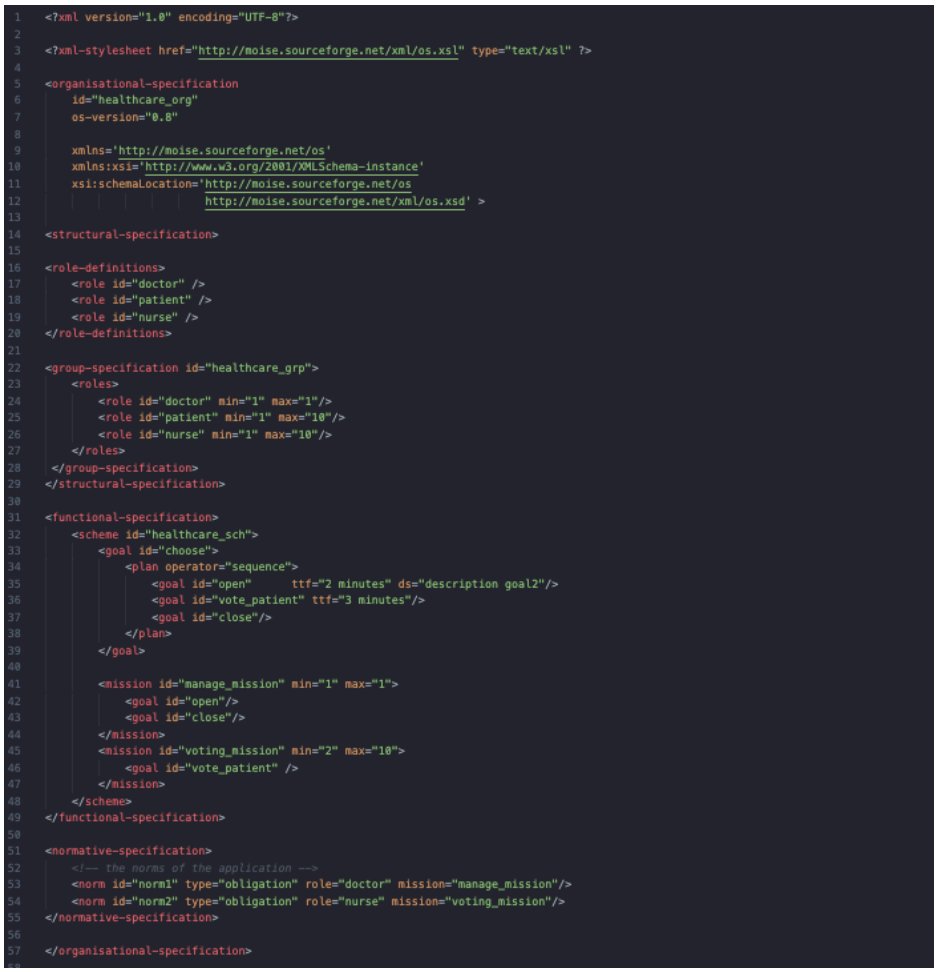
```xml
<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet href="http://moise.sourceforge.net/xml/os.xsl" type="text/xsl" ?>

<organisational-specification
    id="emergency_room_org"
    os-version="0.8"

    xmlns='http://moise.sourceforge.net/os'
    xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xsi:schemaLocation='http://moise.sourceforge.net/os
                        http://moise.sourceforge.net/xml/os.xsd' >

<structural-specification>

<role-definitions>
    <role id="pathological" />
    <role id="normal" />
</role-definitions>

<group-specification id="emergency_room_grp">
    <roles>
        <role id="pathological" min="0" max="1"/>
        <role id="normal" min="1" max="100"/>
    </roles>

    <links>
       <link from="pathological" to="normal" type="authority" scope="intra-group" bi-dir="false"/>
    </links>
    <formation-constraints>
       <compatibility from="normal" to="pathological" />
    </formation-constraints>
</group-specification>
</structural-specification>

<functional-specification>
    <scheme id="emergency_room_sch">
        <goal id="treatment">
            <plan operator="sequence">
                <goal id="tell_condition" ttf="20 minutes" ds="description goal2"/>
                <goal id="treatment_process"/>
            </plan>
        </goal>

        <mission id="telling_mission" min="1" max="1">
            <goal id="tell_condition"></goal>
        </mission>
        <mission id="getting_treatment_mission" min="3" max="3">
            <goal id="treatment_process" />
        </mission>
    </scheme>
</functional-specification>

<normative-specification>
    </properties-->

    <!-- the norms of the application -->
    <norm id="norm1" type="obligation" role="pathological" mission="telling_mission"/>
    <norm id="norm2" type="obligation" role="normal" mission="getting_treatment_mission"/>
</normative-specification>

</organisational-specification>
```

Healthcare Organization (healthcare_org.xml):

The Healthcare Organization, defined in healthcare_org.xml, serves as a foundational structure for orchestrating healthcare-related activities within the Multi-Agent System (MAS).

The organization establishes three primary roles **"doctor", "patient", and "nurse"** each with specified participation constraints. The organizational framework outlines a functional scheme named **"choose"**, which encompasses goals for opening, voting on patients, and closing the process.

Normative specifications introduce obligations that reinforce role-specific responsibilities, such as the **"doctor"** obligation to manage the mission and the **"nurse"** obligation to participate in the patient voting mission. This organization creates a structured environment for collaborative decision-making and patient prioritization, aligning roles and missions to achieve cohesive healthcare management within the MAS.



```xml
<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet href="http://moise.sourceforge.net/xml/os.xsl" type="text/xsl" ?>

<organisational-specification
    id="healthcare_org"
    os-version="0.8"

    xmlns='http://moise.sourceforge.net/os'
    xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xsi:schemaLocation='http://moise.sourceforge.net/os
                        http://moise.sourceforge.net/xml/os.xsd' >

<structural-specification>

<role-definitions>
    <role id="doctor" />
    <role id="patient" />
    <role id="nurse" />
</role-definitions>

<group-specification id="healthcare_grp">
    <roles>
        <role id="doctor" min="1" max="1"/>
        <role id="patient" min="1" max="10"/>
        <role id="nurse" min="1" max="10"/>
    </roles>
 </group-specification>
</structural-specification>

<functional-specification>
    <scheme id="healthcare_sch">
        <goal id="choose">
            <plan operator="sequence">
                <goal id="open"      ttf="2 minutes" ds="description goal2"/>
                <goal id="vote_patient" ttf="3 minutes"/>
                <goal id="close"/>
            </plan>
        </goal>

        <mission id="manage_mission" min="1" max="1">
            <goal id="open"/>
            <goal id="close"/>
        </mission>
        <mission id="voting_mission" min="2" max="10">
            <goal id="vote_patient" />
        </mission>
    </scheme>
</functional-specification>

<normative-specification>
    <!-- the norms of the application -->
    <norm id="norm1" type="obligation" role="doctor" mission="manage_mission"/>
    <norm id="norm2" type="obligation" role="nurse" mission="voting_mission"/>
</normative-specification>

</organisational-specification>
```

System Workflow:

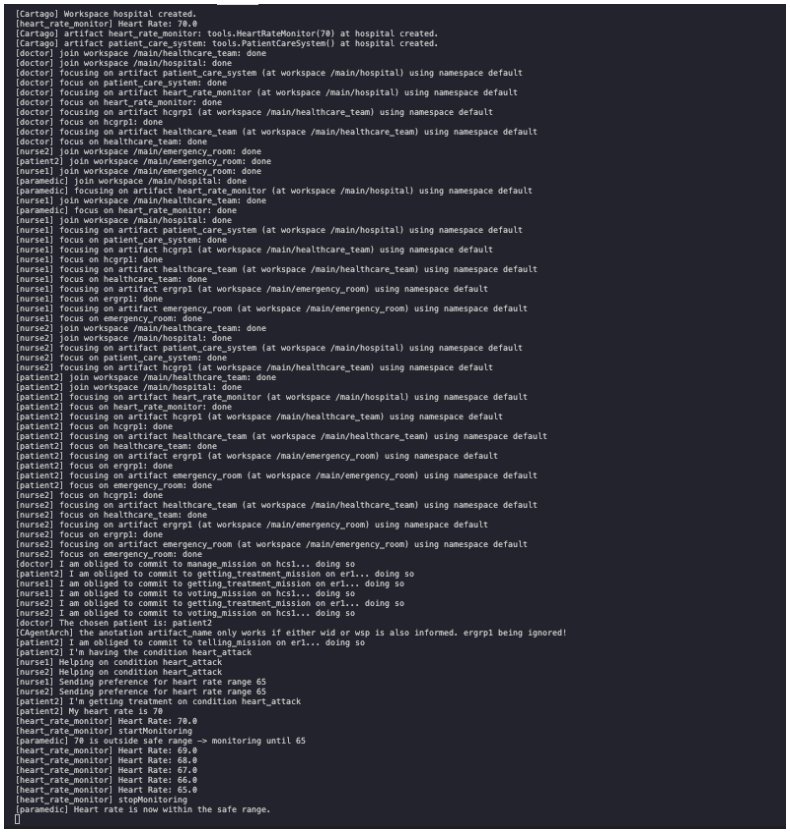
The system workflow is orchestrated through a combination of agent interactions and organizational specifications, ensuring seamless healthcare management within the Multi-Agent System (MAS).

The **PatientCareSystem** artifact manages patient prioritization through a decentralized voting process involving **doctors, nurses, and patients**. The result of this **prioritization** influences decision-making and subsequent patient care.

Meanwhile, the **HeartRateMonitor** artifact simulates and monitors heart rates, with **paramedics** actively engaged in overseeing heart rate changes and taking appropriate actions.

Agent communication is facilitated to share vital information such as medical conditions, preferences, and treatment processes.

The organizational specifications, defined in **emergency_room_org.xml** and **healthcare_org.xml**, guide these interactions, enforcing role-based obligations and maintaining a structured workflow that aligns with predefined rules and goals.



```
mas final {

    // agent definitions
    agent doctor {
        focus: hospital.patient_care_system
               hospital.heart_rate_monitor
    }

    agent paramedic {
        focus: hospital.heart_rate_monitor
    }

    agent nurse1 : nurse.asl {
        beliefs: my_priorities([patient1,patient2])
                 preferred_heart_rate_range(65)
                 recipient_agent(paramedic)
        focus: hospital.patient_care_system
    }

    agent nurse2 : nurse.asl {
        beliefs: my_priorities([patient2])
                 preferred_heart_rate_range(65)
                 recipient_agent(paramedic)
        focus: hospital.patient_care_system
    }

    agent patient1 : patient.asl {
        beliefs: medical_condition("injury")
                 heart_rate(120)
    }

    agent patient2 : patient.asl {
        beliefs: medical_condition("heart_attack")
                 heart_rate(70)
        focus: hospital.heart_rate_monitor
    }

    // artifact definitions
    workspace hospital {
        artifact patient_care_system : tools.PatientCareSystem()
        artifact heart_rate_monitor: tools.HeartRateMonitor(70)
    }

    // hospital organization
    organisation healthcare_team : healthcare_org.xml {
        group hcgrp1 : healthcare_grp {
            responsible-for: hcs1
            players: doctor doctor,
                     nurse1 nurse,
                     nurse2 nurse,
                     patient2 patient

        }
        scheme hcs1: healthcare_sch {

        }
    }

    // emergency or nromal organization
    organisation emergency_room : emergency_room_org.xml {
        group ergrp1 : emergency_room_grp {
            responsible-for: er1
            players: nurse1 normal
                     nurse2 normal
                     patient2 normal

        }
        scheme er1 : emergency_room_sch {

        }
    }

    // agent source path
    asl-path: src/agt
              src/agt/inc

}
```

Main .jcm file

Patient Prioritization:

The core of the system revolves around the prioritization of patients based on their medical conditions. This process is facilitated by the PatientCareSystem artifact, which manages a voting mechanism involving doctors, nurses, and patients.
Agents actively participate in the voting process, expressing preferences and contributing to the decision-making. The Borda count algorithm aggregates these preferences, resulting in a prioritized list of patients.

Heart Rate Monitoring:

The HeartRateMonitor artifact simulates continuous heart rate monitoring. Paramedic agents actively engage in monitoring the heart rates of patients.
If a patient's heart rate exceeds predefined thresholds, the paramedics take appropriate actions.
The monitoring process is dynamic and responds to real-time changes in heart rates.

Agent Interactions:

Agents communicate to share crucial information such as medical conditions, treatment preferences, and heart rate updates. These interactions are guided by the organizational specifications (emergency_room_org.xml and healthcare_org.xml).
The organizational specifications define roles, group structures, and normative rules, ensuring that agents adhere to their responsibilities and obligations within the system.

Decentralized Decision-Making:

The MAS promotes decentralized decision-making, with each agent contributing to the overall decision on patient prioritization. This approach enhances collaboration and leverages the collective intelligence of the system.

Agents, including doctors, nurses, and patients, actively engage in the decision-making process, fostering a distributed and adaptive healthcare management system.

Artifact Integration:

Artifacts such as PatientCareSystem and HeartRateMonitor play a pivotal role in centralizing key functionalities. The PatientCareSystem artifact manages the voting process, while the HeartRateMonitor artifact handles heart rate monitoring.
This modular and integrated approach enhances the scalability, maintainability, and extensibility of the MAS, providing a solid foundation for future enhancements.

Normative Guidelines:

Normative specifications in the organizational structure define obligations for roles in specific missions. For example, the emergency_room_org.xml imposes norms on the "pathological" and "normal" roles, while healthcare_org.xml sets norms for the "doctor" and "nurse" roles.
The adherence to these norms ensures that the MAS operates within predefined rules, promoting ethical and responsible healthcare practices.

Conclusion:

This system presents a comprehensive and decentralized approach to decision-making and patient care. By integrating agent roles, artifacts, and organizational specifications, the system achieves a robust framework that prioritizes patients, monitors critical health parameters, and enforces normative obligations within a dynamic healthcare environment.

The use of Borda count for decision-making, coupled with role-based structures and modular artifacts, contributes to the adaptability and scalability of the system.

The project lays the groundwork for future enhancements, suggesting potential directions such as dynamic role adaptation, real-time monitoring integration, and the exploration of human-agent collaboration.