RESEARCH ARTICLE OPEN ACCESS

Agent Oriented Patient Scheduling System: A Concurrent Metatem Based Approach

Hossainara Begum[1], Shibakali Gupta[2]
[1]Dept. of Computer Science & Engineering, University Institute of Technology, Burdwan, West Bengal, India

Abstract
The Problem of Patient Scheduling[6] is a major issue in a Medical Healthcare System[4]. In India, Healthcare is an 80 billion dollar Industry and is growing at an average rate of 17% annually. However, quality healthcare is still out of reach for many. Each year thousands of fatalities arise simply due to the fact that patient could not be provided with proper medical facilities at the right time. A software agent may be a member of a Multi-Agent System[2][5] (MAS) which is collectively performing a range of complex and intelligent tasks. Using Concurrent Metatem[3], a Multi-Agent Language, we have attempted to model a patient scheduling system[4][6] that can help hospitals collaborate among them through a Liaison-Agent, in order to provide patients with the best care possible. Patients should no longer have to be turned down when hospitals are packed to capacity; instead, they could simply be shifted to another hospital. Hospitals and even doctors are assigned to patients after an automated process of matching patient needs with doctor expertise and hospital infrastructure-leading to reduced waiting-time while maximizing efficiency and potentially saving lives.
***Index Terms*—**Patient Scheduling, Multi Agent System, Concurrent Metatem

I. INTRODUCTION

The Problem of Patient Scheduling[6] is a major issue in a Medical system[10] and its characterized by high certainty and dynamic changes in patient treatment. This scheduling is very complex and concurrency may occur if more than one patient is needed to be scheduling in a same time. If possible then sometimes they are suggested to be transfer with proper information. Patients are to be scheduled to reduce the waiting time and to reduce idle time of resource[1]. This scheduling problem can be solved for multiple departments in Medical System[6]. Clinical/nonclinical department interaction makes inappropriate patient transfer. Clinical/clinical department interaction increases the waiting time of the patient in a department. In health department, patients can undergo any checkup. When a particular resource gets overcrowded the patients can go any free resources because there is no constraint. New method has been proposed in this paper to transfer the patients from overcrowded resources to free resources. To optimize its goal we have used a Multi Agent based patient scheduling technique[4][6]. Multi agent system[2] can be used to solve this problem which is difficult or impossible for an individual agent or a monolithic system to solve. Multiple agents share the messages to interact with each other. Multi Agent System[2] change the complex problem into simple which can be solved easily. Multi Agent System[2] is a system composed of multiple interacting intelligent agents[2], is used to solve many problems that are difficult or impossible for an individual agent to do. In medical[10] expert system they simplify complex problem solving by dividing the necessary knowledge into few subunits to which an independent intelligent agent[2] is associated and thus the agents' activity will be so coordinated. In this way we can refer Multi Agent System[2]. A Concurrent Metatem[3] system contains a set of concurrently executing agents which can communicate with each other via asynchronous broadcast message passing. Here we have tried to implement a Multi Agent Based Patient Scheduling[4][6] by using Concurrent Metatem[3] approach. A Multi Agent System[2] is a computer based system, used to solve several kinds of problem that are quite difficult and impossible for an individual to solve. It is composed of multi interacting intelligent agents[2] within an environment. Multi-Agent Systems[2] consist of several agents and their several environments. Multi Agent Systems research refers to software agents **[**2]. The agents in a Multi Agent System[2] could equally to be robots, humans or human teams. A Multi-Agent System[2] may contain combined of human-agent. The medical expert system[10] agents can perceive from environment and interact with the environment by learning and executing different action on the environment autonomously. They can communicate with each other that allow co-operative problem solving[1]. Multi Agent System[2] can be applied to Artificial Intelligence[2] [9] which divide all the necessary knowledge or information into subunits to which an independent intelligent agent[2] [10] is associated to act. They can solve problem by coordinating the agents' activity. The aim of this

system is to recognize how important processes can be coordinated. In Multi Agent System[2] an agent [12] is a computerized entity like a computer programme or a robot. An agent can interact autonomously because it has the capacity to adopt with environment changes. A Multi Agent System[2] is composed of a set of computer processes exist at the same time, communicate with each other in an environment.

II. PATIENT SCHEDULING COORDINATION ARCHITECTURE

Patient scheduling in Multi agent System[2][6] process is developed for dynamic load balancing in agent based simulation. Each agent executes their action by message communication with others. Based on this scheduling method, we have tried to schedule all the patients (concurrently) by using Multi Agent System[2]. This method provide patient the best care possible. This method reduces the waiting time of the patient and improves the patient satisfaction. For better treatment Patients should be shifted to another resources. We have designed the architecture of patient scheduling, is shown below:

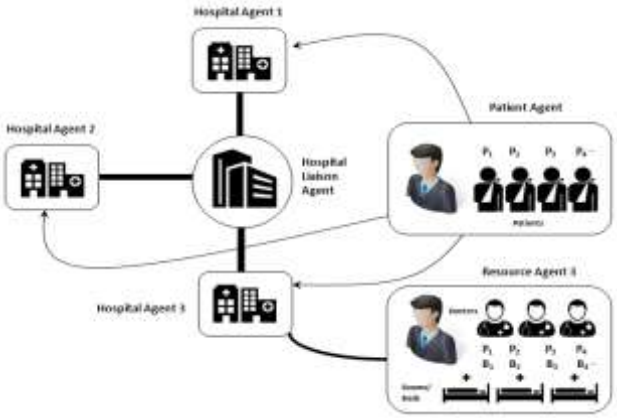

Fig. 1.Architecture of the proposed patient scheduling

From the above architecture, there are three types of Agents are used:

1. *Hospital Agent (HA):* Acts on behalf of a Hospital. Each hospital has a Hospital Agent which is responsible for Patient Admission.
2. *Patient Agent (PA):* Acts on behalf of a Hospital. Each hospital has a Patient Agent which handles multiple patients and allots rooms and doctors to them.
3. *Hospital Liaison Agent (HLA):* Central Agent which is responsible for hospital communication and patient shifts between them.

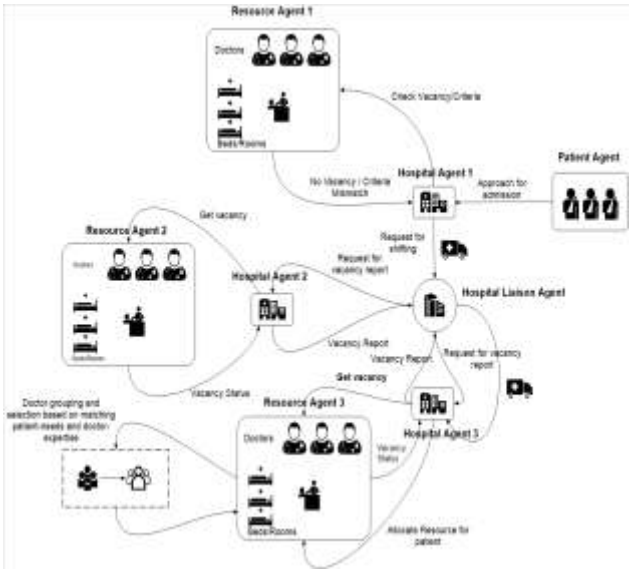

Fig. 2.Architecture of the Agent communication

Hospital Liaison Agent (HLA) is in charge of agent communication. It can receive Resource/Hospital agent information, process the patient agent. It is a type of Central Agent and also maintains the count of patients to be shifted. The Patient Agent can handles multiple patient and provide proper medical care based on their requirements. At each resource/Hospital there is one Patient Agent which is in charge of patient scheduling procedure[1], selecting the Patient sets based on some criteria and moving them. Resource/Hospital Agents send and receive the requests from Hospital Liaison Agent. When a particular resource/Hospital gets overcrowded or if Patients require better treatment to be alive, it needs to migrate the exceeding number of patients to another resource in order to reduce the waiting time of the patient and to improve the resource utilization for Patient Satisfaction. The overcrowded resource will send request to neighborhood resource via Hospital Liaison Agent which response if it has enough space to admit that patient. Moreover, rooms are allotted and available doctors are assigned to them by finding their needs through an automated process. And If not then it will ignore the request[1].

III. THE PROPOSED METHOD OF PATIENT SCHEDULING

A. Step1: Procedure Patient Admission-

1. Input Patient Details like name, sex, age, history, department, status and criteria.
2. Check if there is any vacancy in the Current-Hospital for the given department and criteria.
3. If there is no vacancy, go to step 4 else go to step 5.
4. Attempt to transfer the patient to another hospital, and go to step 6.
5. Allot hospital resources to the concerned patient.

6. Stop.

B. Step2: Procedure GetVacancyStatus-

1. Input Hospital-Name, Patient-Department and Patient-Criteria.
2. Check the number of vacant rooms in the given department at the given hospital.
3. If the number of vacancies is 0, then return FALSE.
4. If the number of vacancies is more than 0, check whether the patient's criteria and the hospital's facilities match or not.
5. If they do match, return TRUE, else return FALSE.
6. Stop.

C. Step3: Procedure GetVacancyCount-

1. Input Hospital-Name and Patient-Department.
2. Initialize C to 0.
3. For Each Room in the given Hospital, do step 4.
4. If the selected room is not booked AND the selected room belongs to the given department, then increment C by 1.
5. Return C.
6. Stop.

D. Step4: Procedure CriteriaMatch-

1. Input Hospital-Name, Criteria and Patient-Department
2. If the given hospital does not have the given department, then return FALSE.
   For each specific_hospital_criteria and
3. For each Specific_Hospital_Criteria and Specific_Patient_Criteria In Hospital_Facilities and Patient_Criteria, do steps 4-5.
4. Compute P=ABS(Specific_Hospital_Criteria - Specific_Patient_Criteria).
5. If P > CRITERION_THRESHOLD (for that specific criteria) then return FALSE.
6. Return TRUE.
7. Stop.

E. Step5: Procedure ShiftPatient-

1. Input Patient-Name, Sex, Age, Patient-History, Patient-Criteria, Patient-Department.
2. Initialize Proposed_Hospitals to NULL.
3. For each Hospital IN the list of all registered hospitals, do step 4.
4. If vacancy status at the selected hospital with respect to the patient-criteria is TRUE, then add the selected hospital to the set of proposed hospitals.
5. If Proposed_Hospitals is NULL, then return FALSE.
6. Initialize BestHospital to NULL and DepartmentRank to 0.
7. For each Hospital in Proposed_Hospitals do steps 8-9.
8. Set R = Department Rank of the selected hospital and given patient-department.
9. If R > DepartmentRank, set DepartmentRank to R and BestHospital to Selected Hospital.
10. Allocate resources to the patient in the BestHospital.
11. Return TRUE.
12. Stop.

F. Step6: Procedure AllotResourceToPatient-

1. Input Name, Sex, Age, Patient-History, Criteria, Patient-Status, Department.
2. Allot new patient ID to the patient.
3. Allot new room to the patient.
4. Allot new doctor to the patient.
5. Record ID, Room, Doctor, Patient-Details onto database.
6. Stop.

G. Step7: Procedure AllotRoom-

1. Input Age, EmergencyLevel, Status, Department.
2. If Status=Discharged then return NULL.
3. If Status=Surgery then return the first available OT.
4. If Department=Unallocated then go to step 5 else go to step 11.
5. If Status=General_Admission then return a vacant Room in the "General" department.
6. If Status=Emergency or Status=In_Recovery then go to step 7.
7. If Age<=1 then return a vacant Room in the "NICU" department.
8. If Age>1 And Age<=10 then return a vacant Room in the "PICU" department.
9. If Age>10 And EmergencyLevel>=9 return a vacant Room in the "CCU" department.
10. If Age>10 And EmergencyLevel>=7 And EmergencyLevel<9 then return a vacant Room in the "ICU" department.
11. Return a vacant Room in the given department.
12. Stop.

H. Step8: Procedure AllotDoctor-

1. Input EmergencyLevel, Status, Department.
2. Initialize BestMatch=0, BestDoctor=NULL.
3. Initialize AvailableDoctors = Set of doctors in the given department And who are available on-call.
4. For each Doctor in AvailableDoctors do steps 5-6.
5. Set MatchScore=DoctorPatientScore(Age, Emergency, Status, Doctor).
6. If MatchScore>BestMatch then set BestMatch=MatchScore and BestDoctor=Doctor.
7. Return BestDoctor.
8. Stop.

I. Step9: Procedure GetDoctorPatientScore-

1. Input Age, EmergencyLevel, Status, Doctor.
2. Retrieve from database the Specialization, EmergencyExpertise and AgeSpecialization of the given doctor.
3. If AgeSpecialization=Paediatric and Age>=18 then return 0.
4. Initialize Score=10*(EmergencyExpertise - EmergencyLevel).
5. If Specialization=Surgery then go to step 6 else go to step 7.
6. If Status=Surgery then Increment score by 10 else Decrement score by 10.
7. If Specialization=Diagnosis then go to step 8 else go to step 9.
8. If Status=Diagnosis then Increment score by 10 else Decrement score by 10.
9. Increment score by 10.
10. Return score.
11. Stop.

IV. PROPOSED ALGORITHM

PATIENT/HOSPITALCRITERIA(for EACH criteria, one CRITERION THRESHOLD will be taken):

1. Finance (Value Range: 0-1)
2. Emergency (Value Range: 0-1)

Algorithm:-

J. Step1: Procedure PatientAdmission-

```
Patient Admission (Name, Sex, Age, Patient History, Criteria, Patient Status, Patient Dept) begin
Has Vacancy=Get Vacancy Status(this Hospital, Criteria, Patient Dept);
        if (has Vacancy=false) then
        RequestPatientShift(Name,Sex,Age,Patient History,Criteria,PatientStatus,PatientDept);
        return;
        End if
        AllotResourceToPatient(Name,Sex,Age,PatientHistory, Criteria, PatientStatus, PatientDept);
End
```

K. Step2: Procedure GetVacancyStatus-

```
GetVacancyStatus(Hospital,PatientCriteria,PatientDept) begin
    X=GetVacancyCount(Hospital,PatientDept);
            if(X=0)
        then return 0;
if(CriteriaMatch(Hospital,PatientCriteria,PatientDept)=true)  then return true; return false;
End
```

L. Step3: Procedure GetVacancyCount-

```
GetVacancyCount(Hospital,Dept)
begin
    C=0;
    for each Room in Hospital do begin
    if(Room.Booked=false And Room.Deptartment=Dept) then C=C+1
    end if
    end
    return C;
end
```

M. Step4: Procedure CriteriaMatch-

```
CriteriaMatch(Hospital,Criteria2,PatientDept) begin
if (Hospital.hasDepartment(PatientDept)=false) then return false;
  for each (X,Y) in (Criteria1,Criteria2) do begin
  if (ABS(X-Y)> CRITERION_THRESHOLD) then return false;
  end
return true;
end;
```

N. Step5: Procedure ShiftPatient-

```
ShiftPatient(Name,Sex,Age,PatientHistory,Criteria,PatientDept) begin
  ProposedHospitals=NULL;
  for each Hospital in AllHospitals do begin
  if(GetVacancyStatus(Hospital,Criteria)=true) then
    ProposedHospitals=ProposedHospitals  Hospital;
  end if
end
if(ProposedHospitals=NULL) then return false;
  bestHospital=NULL,deptRank=0;
  for each Hospital in ProposedHospitals do begin
  R=getDepartmentRank(Hospital,PatientDept);
  If(R>deptRank) then
    deptRank=R;
    bestHospital=Hospital;
  end if
end
bestHospital.AllotResourceToPatient(Name,Sex,Age,PatientHistory,Criteria,PatientStatus,PatientDept);
return true
end
```

O. Step6: Procedure AllotResourceToPatient-

```
AllotResourceToPatient(Name,Sex,Age,PatientHistory,Criteria,PatientStatus,PatientDept) begin
Room=AllotRoom(Age,Criteria,Emergency,PatientStatus,PatientDept);
Doctor=AllotDoctor(Age,Criteria,Emergency,PatientStatus,PatientDept);
record Patient, Room, Doctor, Name, Sex, Age, PatientHistory, Criteria, PatientStatus;
end;
```

P. Step7: Procedure AllotRoom-

```
AllotRoom(Age,Emergency,Status,Department)
begin
```

```
   if(Status=DISCHARGED) then return NULL;
   if(Status=SURGERY) then return getFreeOT();
   if(Department=UNKnown)then
      if(Status=EMERGENCY OR
Status=RECOVERY) then      if(Age is NE)then
            return Room where Room.Booked=false
&&                Room .Dept=”NICU”;
            else if(Age IS CHILD) then
            return Room where Room.Booked=false
&&                Room.Dept=”PICU”; else
            if(Emergency is VERY_HIGH) then
                  return Room where
Room.Booked=false &&
   Room.Dept=”CCU”;else
                  return Room where
Room.Booked=false &&
   Room.Dept=”ICU”; end if
            end if
         else if(Status=GENERAL_ADMISSION)then
               return Room where
Room.Booked=false&&
   Room.Dept=”GENERAL”;
      end if
      else

   if(Status=UNDER_SPECIALIZED_TREATMEN
T)          then
            Return Room where Room.Booked=false
&&          Room.Dept=Department;
      end if
   end if
end;
```

Q. *Step8: Procedure AllotDoctor-*

```
AllotDoctor(Emergency, Status, Department)
begin
bestMatch=0, bestDoctor=NULL;
AvailableDoctors=get doctors from database where
Dept=Department;
for each Doctor in AvailableDoctors do begin
   matchScore=GetDoctorPatientScore(Age,Emergen
cy,Statu  s,Doctor);
   if(matchScore>bestMatch) then
         bestMatch=matchScore;
         bestDoctor=Doctor;
   end if;
end;
return bestDoctor;
end;
```

R. *Step9: Procedure GetDoctorPatientScore-*

```
GetDoctorPatientScore(Age, Emergency, Status,
Doctor)
begin
Specialization,                EmergencyExpertise,
AgeSpecialization = get record of Doctor;
if(AgeSpecialization = Pediatric AND Age > 18)
return 0;
score=10*(EmergencyExpertise-Emergency);
if(Specialization = Surgery) then
      if(Status=4) then
            score=score+10;
      else
            score=score-10;
      end if
else if(Specialization = Diagnosis) then
      if(Status=2) then
            score=score+10;
      else
            score=score-10;
      end if
else
      score=score+10;
end if
      return score;
end;
```

V. TRANSACTIONAL ANALYSIS THROUGH TIME SEQUENCE DIAGRAM

Time Sequence Diagram provides the way of concurrent process. Hospital Liaison Agent can establish a connection among many resources. when a particular resource agent gets overcrowded then that will send request to another neighborhood resource agent via Hospital Liaison Agent. It will check whether it has enough capacity to do treatment for emergency patients and also check whether rooms are available or not, the required doctors are there or not. If all those criteria become matched it will accept. If not then it will reject the request from overcrowded resource agent. Patients are shifted based on some criteria. Patients are transferring from overcrowded resource agent to neighborhood resource agent. Patients will be moved to another resource agent. Then migrated agent will be removed from overcrowded resource agent. This process will be repeated until the entire overcrowded patient is scheduled. This multi agent based Patient Scheduling[4][6] method has been used to search for optimized scheduling scheme[5], including migrated agent sets. Only a single Hospital Liaison Agent (HLA) exits in the entire system, responsible for Inter Hospital Communication. In this Diagram this selected Hospital Agent simulates the behavior of all Hospital Agent in the System. Messages sent to this Agent are intact symbolic of sending the message to every Hospital Agent. Only the Admit Patient message is sent to a single Hospital Agent which was selected by the Hospital Liaison Agent (HLA) for transferring the Patient. Thus, the selected Patient Agent refers to the Patient Agent of the hospital which was selected by the Hospital Liaison Agent (HLA) for transferring the Patient[1].

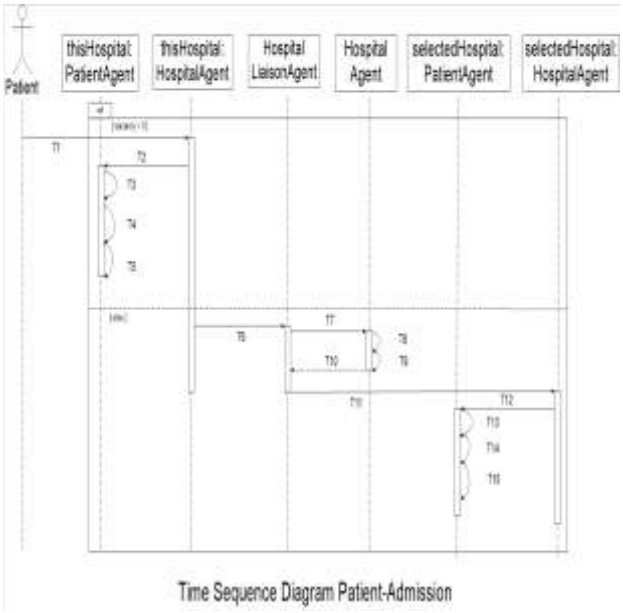

Fig. 3.Time Sequence Diagram

Above diagram describes the time sequence diagram of a complete patient scheduling processing medical system. One patient is to be admitted. Here,

T_1- patient Approach for admission to this Hospital Agent.

T_2- hospital Agent asks to Patient Agent for Resource Allotment of that Patient.

T_3- Patient ID provided/allotted by Patient Agent.

T_4- Room provided/allotted by Patient Agent.

T_5- Doctor provided/allotted by Patient Agent.

T_6- this Hospital Agent request to Hospital Liaison Agent for Patient Shifting.

T_7- Hospital Liaison Agent request for Vacancy report to other Hospital Agent

T_8- Hospital agent search to get Vacancy status.

T_9- Hospital agent search criteria of patient.

T_{10}- Hospital agent provides a vacancy report to the Hospital Liaison Agent.

T_{11}- based on the vacancy report, Patient are requested to admit.

T_{12}- Hospital Agent request to Patient Agent for resource allotment of that Patient.

T_{13}- Patient Agent of the Selected Hospital generates a Patient ID.

T_{14}- Patient Agent of the Selected Hospital Provide/allot room.

T_{15}- Patient Agent of the Selected Hospital provide/allot Doctor.

VI. CONCURRENT METATEM

Concurrent MetateM[2][3] provides an operational framework[9] through which societies of MetateM[7][8][9] processes can operate and communicate with each others. the notion of executing a logical specification which generates individual agent behavior[3]. The temporal connectives [7][8][9] can be divided into two categories which are as follows:

1. Some Strict past time connectives: ● (weak last), □ (Strong Last), ◊ (was), ■ (heretofore), □ (Since) and Z (Zince or weak since)[3][7][8][9].
2. Some Present and future time connectives: O (next), ◊ (sometime), □ (always), U (until) and W (unless).

Here whenever one job is executed the antecedent part will always be about past time and the consequent part will always be about present and future time[3][7][8][9].

For Examples-

1. □important (agents) means "it will always be true that agents are important"[3][7][8][9].
2. ◊ Important (Process) means "sometime in the future, Process will be important"[3][7][8][9].
3. ♦ Important (Prolog) means "sometime in the past it was true that Prolog was important"[3][7][8][9].
4. (¬friends (us)) U apologize (you) means "we are not friends until you apologize"[3][7][8][9].
5. O apologize (you) means "tomorrow (in the next state), you apologize"[3][7][8][9].

VII. IMPLEMENTATION USING CONCURRENT METATEM APPROCH

A. Step1:

Patient-agent (allotResourceForPatient) [patientAdmitted]:
Start => waiting
True => ◊ waiting
□ allotResourceForPatient(Name,Sex,Age,History,Criteria,Status,Dept)=>□allotRoom(HospitalName,Dept,Criteria,Status,Age)
□ allotRoom(HospitalName,Dept,Criteria,Status,Age)=>¬ waiting
□ allotRoom(Hospital,Name,Dept,Criteria,Status,Age)=>□□allotDoctor(HospitaName,Dept,Criteria,Status,Age)
□ allotDoctor(Hospital,Name,Dept,Criteria,Status,Age)∧doctor(Hospital,X,Dept,Criteria) => □ PatientAdmitted(Name,X)
□ allotDoctor(HospitalName,Dept,Criteria,Status,Age)=>□□waiting.

B. Step2:

hospital-liaison-agent(shiftPatient,vacancyReport) [hasVacancy,admitPatient]:
start => known(hospital-agent)
□ shiftPatient(Name,Sex,Age,History,Criteria,Status,Dept)∧known(H)∧hospital(H,C,Criteria)=> □ ¬ waiting
□ shiftPatient(Name,Sex,Age,History,Criteria,Status,Dept)∧known(H)∧hospital(H,C,Criteria)=>□ hasVacancy(Name, Sex,Age,History, Criteria,Status, Dept)
□ vacancyReport(N)∧(0<N)∧¬waiting=>□ admitPatie

nt(Name Sex, age, History, criteria Status,Dept)
□vacancyReport(N) ∧ (0<N) ∧ ¬waiting =>□ waiting

C. Step3:

hospitalagent(admitPatient,patientAdmitted,dischargePatient,hasVacancy)
[shiftPatient,vacancyReport]:
□ admitPatient(Name,Sex,Age,History,Criteria,Status,Dept)∧vacancy(P)∧(0<P)=>□ □ servePatient(Name,Sex,Age,History,Criteria,Status,Dept)
□ admitPatient(Name,Sex,Age,History,Criteria,Status,Dept)∧vacancy(0)=>□ □ ShiftPatient(Name,Sex,Age,History,Criteria,Status,Dept)
□ servePatient(Name,Sex,Age,History,Criteria,Status,Dept)∧vacancy(X)∧ (Y=X-1)=>□ vacancy(Y)
□ servePatient(Name,Sex,Age,History,Criteria,Status,Dept)=>□ allocateResourceForPatient(Name,Sex,Age,History,Criteria,Status,Dept)
□ dischargePatient(Name,Sex,Age,History,Criteria,Status,Dept)=> vacancy(X)∧ (Y=X+1) => □ vacancy(Y)
□ hasVacancy(Name,Sex,Age,History,Criteria,Status,Dept)∧department(self,Dept)∧vacancy(X)∧(0<X)=> □ vacancyReport(1)
□ hasVacancy(Name,Sex,Age,History,Criteria,Status,Dept)∧vacancy(0) =>□ vacancyReport(0)
□ hasVacancy(Name,Sex,Age,History,Criteria,Status,Dept)∧department(selfDept) => □ vacancyReport(0).

VIII. EXPERIMENTAL RESULTS

To implement this scheduling method for distributed patients with grouping, hardware requirements and software requirements are used. Hardware Requirements are Intel® Core (TM) i3-4030U CPU @ 1.90 GHz processor and 64- bit operating system are used. Software Requirements are JDK 1.6 java software used with METATEM package. Three resources and Ten patient Agents are created to do the scheduling. Initially create the patient agent by retrieving from database. Each patient agent have unique agent id which will be generated while creating an agent. Created patient agent should be send to resource agent to do their treatment. Through agent communication one resource agent sends the message to another resource agent. After receiving the responses, patient agents are moved from one resource agent to other resource agent. Available rooms are allotted and Available doctors are also be assigned to them. Thus the scheduling has been done and minimizes the fatalities of a patient lives.

IX. CONCLUSION

In this paper we have framed a novel Multi Agent based patient scheduling[6] algorithm and we have implemented it through MetateM language. In this algorithm we have tried to execute all the processes in a concurrent way. And to hold the rhythm we introduced Concurrent MetateM[2][3] approach. But in future we can use any other novel (customized) techniques for better concurrency. Color Petri Net (CPN) concept can be an alternative in this context.

REFERENCES

[1.] G. Mageshwari, E. Grace Mary Kanaga, Department of Computer Science and Engineering, Karunya University, Coimbatore, India. magesh.5.spgm@gmail.com, grace@karunya.edu, “A Distributed Optimized Patient Scheduling using Partial Information”.

[2.] Michael Wooldridge Department of Computer Science, University of Liverpool, Uk', " An Introduction to Multiagent Systems".

[3.] Michael Fisher, Department of Computing, Manchester Metropolitan University, Manchester M1 5GD, United Kingdom, M.Fisher@mmu.ac.uk “A suvey of Concurrent MetateM-the language and its application”.

[4.] Antonio Moreno , "Medical Applications of Multi-Agent Systems" Computer Science & Mathematics Department, Universitat Rovira i Virgili ETSE. Campus Sescelades. Av. dels Països Catalans, 26, 43007-Tarragona, Spaina moreno@etse.urv.es

[5.] Yoav Shoham, Stanford University, Kevin Leyton-Brown, University of British Columbia "MULTIAGENT SYSTEMS Algorithmic, Game-Theoretic, and Logical Foundations".

[6.] Anja Zöller, Lars Braubach, Alexander Pokahr, Franz Rothlauf, Torsten O. Paulussen, Winfried Lamersdorf , Armin Heinzl, Department of Business Administration and Information Systems, University of Mannheim, D-68131 Mannheim, Germany Email: {zoeller; rothlauf; paulussen; heinzl}@uni-mannheim.de Department of Computer Science, Distributed and Information Systems, University of Hamburg, D-22527 Hamburg, Germany Email: {braubach; pokahr; lamersdorf}@informatik.uni-hamburg.de, "Evaluation of a Multi-Agent System for Hospital Patient Scheduling".

[7.] Michael Wooldridge, Mitsubishi Electric Digital Library Group, 18th Floor, Centre Point, 103 New Oxford Street, London WC1 1EB, United Kingdom, mjw@dlib.com, "A Knowledge-Theoretic Semantics for Concurrent Metatem".

[8.] H. Barringer, M. Fisher, D. Gabbay, G. Gough and R. Owens, Department of Computer Science, University of Manchester, Manchester, Department of Computing, Manchester Metropolitan University, Manchester, Department of Computing, Imperial College of Science, Technology and Medicine, London, Nomura Research Institute Europe Ltd., London, "Metatem: An Introduction".

[9.] Marco Alberti, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, "Agent-Oriented Programming".

[10.] Antonio Moreno, Computer Science & Mathematics Department, Universitat Rovira i Virgili, ETSE. Campus Sescelades. Av. dels Països Catalans, 26, 43007-Tarragona, Spain, amoreno@etse.urv.es, "Medical Applications of Multi Agent Systems.