Proceedings of the 9th International Conference on Applied Informatics
Eger, Hungary, January 29–February 1, 2014. Vol. 1. pp. 131–140
doi: 10.14794/ICAI.9.2014.1.131

The GAIA methodology - A case study based on the NEXT-TELL project

Roland Sipos

Eötvös Loránd University Budapest, HU
Roland.Sipos@cern.ch

Abstract

The GAIA[1] methodology deals with the macro and micro-level analysis and design of multiagent-agent systems, that focuses on the computational organisation between interacting roles. I will illustrate a case study for the design of a unique system that is based on this methodology.
NEXT-TELL[2] is an Integrated Project with the main objective to provide computational and methodological support for teachers and students, in order to bring the visions of future into todays' classrooms. The different stages and theoretical considerations of the project can be transparently modeled by the GAIA multiagent methodology. The NEXT-TELL environment is relatively open, and highly dynamic that cannot be easily modeled with standard object oriented techniques, but the autonomic system nomenclature naturally fits to the project's terminology and the different layers of ECAAD can be modelled with the GAIA phases[3]: analysis, architectural and the detailed design.
In this contribution I will introduce a case study for the design of a NEXT-TELL like system that is based on GAIA elements and techniques.

Keywords: GAIA methodology, multi-agent, NEXT-TELL

1. Introduction

The NEXT-TELL classroom is a perfect example for a case where the system can be modelled with the multi-agent paradigm. The environment is relatively open, and highly dynamic, that cannot be easily modeled with standard object oriented techniques. Participants' behavior is not a static notion therefore may be described as autonomous actions in the environment. The autonomic system nomenclature naturally fits or can be matched to the NEXT-TELL system terminology. The different NEXT-TELL layers can be followed through as the evolvement of the GAIA sequence: analysis, architectural and the detailed design.

2. Analysis

Suppose that we got the same **requirements** as the NEXT-TELL system. The first step includes the discovery of possible sub-organization in the global scope. The NEXT-TELL project contains a well detailed methodology for a creation of learning materials and artifacts based on the ECAAD method. ECAAD deploys materials to the environment of the multi-agent system that will be manipulated and used by the agents themselves. We can discover at least the two following **sub-organizations** in the global system:

- **ECAAD Engineering**: The goal of this organization is the provision of ICT support for the virtual classroom, including development of new learning materials and the support of assessment techniques for teachers.
- **ECAAD Using**: This sub-organization will use the materials, controlled by the Cloud controllers and ECAAD engineers. The aim of this group is different, as the agents will manipulate the available environmental structures independently from the creation and control on these modules.

Before we can proceed to the collect the possible roles and interactions focused on each sub-organization, one need to define the **environmental model**. The learning environment has to be threaten as an abstract resource, therefore the following possible elements will be defined and discussed later (figure 1):

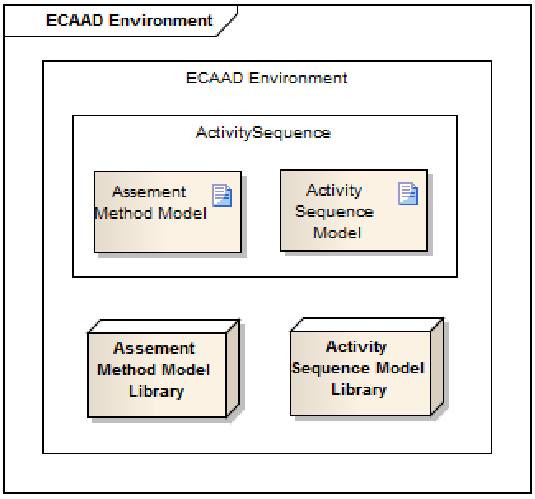

Figure 1: ECAAD environment base for the agents.

- **Activity model**: A given learning activity model definition, with a method that delivers development instructions for the actual learning environment.

- **Assessment model**: The assessment term is closely related to the linked activity itself, that describes the general satisfactory attribute of a given activity model.
- **Libraries**: Storage elements of the models above.
- **Activity sequence**: The product of an activity and assessment model pair.

The preliminary role model defines the basic attributes and functionality to solve the given task in the agent environment. Roles are defined with four attributes (permissions, responsibilities, activities and protocols), and the standard role template from F. Zambonelli [1] is used to define our roles in a NEXT-TELL system. The ActivityPlanner role is missing due to that role is only a bridge between the ActivityLeader and ECAAD Tools.

Table 1: Preliminary role ECAAD Engineer

Role schema	**ECAAD Engineer**
Description	This preliminary roles involved in method particularization, implementation and deployment.
Protocols and activities	$\underline{MethodParticularisation}$, $\underline{MethodImplementation}$, $\underline{MethodDeployment}$
Permissions	**Modeling** *Activities and Assessments* **Implement** *Activity and Assessment Models* **Deploy** *Activity and Assessment Tools and Models*
Liveness	ECAAD Engineer = (**MethodParticularisation.** **.MethodImplementation** **.MethodDeployment**)*
Safety	$numberOfDeployed > 0$

Table 2: Preliminary role ECAAD Planner

Role schema	**ECAAD Planner**
Description	Involved in the selection of given ECAAD models, based on the activity plan that is a draft of required learning activities.
Protocols and activities	$StartAct$, $ProvActSetup$, $\underline{PrepAct}$, $\underline{SaveRes}$, $\underline{Monitor}$
Permissions	**Setup** *ECAAD UserInterface* **Update** *ECAAD UserInterface*

Liveness	ECAAD Planner = (**ProvPlan**.SetupAct.HandleAct)$^{\omega}$
	SetupActivity = **ProvActSetup**.$\underline{\text{PrepAct}}$.**StartAct**
	HandleActivitiy = ($\underline{\textbf{Monitor}}$ ‖ $\underline{\text{SaveRes}}$)*

Safety	$numOfTraced \wedge numOfMonitored >= numberOfSetUp$

Table 3: Preliminary role ECAAD UserInterface

Role schema	**ECAAD UserInterface**
Description	Involved in the control of learning activity, including the guidance of other participants in the collaboration.
Protocols and activities	*UISetup*, *ProvActAccess*, *StopAct*, *FeedbReq*, *GiveGuide*, *SendFeedb*
Permissions	**Setup** *Models through ECAAD Planner* **Monitor** *Models through ECAAD Planner* **Trace** *Models through ECAAD Planner* **Update** *Models through ECAAD Planner*

Liveness	ECAAD UI = (MakeUI.Actions.EndActivity)$^{\omega}$
	MakeUI = (**UISetup**.**ProvActAccess** $\parallel$ [**GiveGuide**])*
	EndActivity = (**StopAct**.**FeedbReq**.**SendFeedback**)*

Safety	$numOfTraced \wedge numOfMonitored >= numberOfSetUp$

Table 4: Preliminary role ECAAD User - Student

Role schema	**ECAAD User - Student**
Description	Involved in the learning in the current activities, working with the available tasks and tools.
Protocols and activities	*EngageWithAct*, *GetFeedb*, *ReqGuide*, *NoteGuide*, *DescRes*, *GetGuide*, *EndAct*, *ProgressInAct*
Permissions	**Use** *Tools and Applications*

Liveness	ECAAD Student = (**EngageWithAct.[AskGuide] .ProgressInAct.FinishActivity**)*
	AskGuide = (**ReqGuide.NoteGuide**)*
	FinishActivity = **EndAct.GetFeedb.DescRes**

Safety	$toolsAreReadyToUse = TRUE$

Table 5: Preliminary role ECAAD User - ActivityLeader

Role schema	**ECAAD User - ActivityLeader**
Description	Involved in the leading of learning exercises with the already loaded learning model.
Protocols and activities	*GiveFeedb*, *MonitorProgress*, *Instruct*, *ForcedEndActivity*
Permissions	**Use** *Tools and Applications*
Liveness	ActivityLeader = (**Instruct.MonitorProg.GiveFeedb**)$^{+}$ $\parallel$ [ForcedEndActivity]
Safety	$toolsAreReadyToUse = TRUE$ $numOfParticipants >= 0$ $numOfFeedback == numOfParticipants$

In the preliminary roles, permissions are unique methods of a given role that describes manipulation of environmental resources and artifacts. We usually use permissions too in the case when roles affect other role's resources. In this study I give a textual description for the role's permissions.

- **Modeling, Implement, Deploy** - The ECAAD Engineer has these permissions in order to create and place activity and assessment ECAAD models into the global repository, therefore making them available for usage.
- **Setup, Update, Monitor, Trace** - The ECAAD Planner and UserInterface has these permissions. The Planner can setup and update elements and resources of the UserInterface, and that is allowed to trace and control learning activities from the ECAAD Environment, but only through the controller Planner role.
- **Use** - This common permission is possessed by the ECAAD users, namely by the student and activity planner and leader roles. This permission expresses that the role may control elements of other roles that are allowed by the resource holder.

The next step of the analysis phase is to describe the **interaction model**, that consists the iteration of protocol identification and role model elaboration. Protocol definitions usually shown as diagrams, and can be found in the study[4]. To finish the analysis phase the **organization rule** definitions are presented as follows.

1. **ECAAD Engineering sub-organization**: Responsible to ensure the OLM availability in the databases. For this, they have one general role, that may be filled several times.
   (a) Liveness rules ensure continuously development of new OLMs, and ECAAD tools.
      - $Particularisation^x \rightarrow Implementation^x \rightarrow Deployment^x$ - The ECAAD Engineer's activities may be continued in a given sequence.

   (b) Safety rules of this organization need to ensure the OLM container integrity, and that the single role can be filled multiple times.
      - $ECAADEngineer^{1..n}$ - At least one developer, with maximum n.
      - $Particularisation[i] \rightarrow Implementation[i]$
      - $Implementation[i] \rightarrow Deployment[i]$ - Ensures the correct sequence of the development.
2. **ECAAD Using sub-organization**:
   (a) Liveness rules define the evolvement of learning activities in the classroom.

(b) Safety rules are defined in order to keep strict hierarchy between role instances.

- $\neg((ECAADPlanner \vee ECAADUI) \mid (ActivityPlanner \vee ActivityLeader) \mid Student)$
  The given roles may be not filled in the same time.
- $ActivityPlanner^1$ $ActivityLeader^1$ $Student^{1..n}$ $ECAADPlanner^1$ $ECAADUI^1$
  There must be exactly one teacher and ECAAD tool, as they will drive at least 1 and maximum n students through the learning activity.

3. **Global organization**: Global organizational liveness rules will be seen on illustrations later.

(a) Safety: The global safety rules are express that roles are filled in order to start any cooperation.

- $numberOfSubOrg(ECAADEngineering) >= 1$
  At least one engineering sub-organization exists.
- $numberOfSubOrg(ECAADUsing) >= 1$
  At least one ECAAD using sub-organization exists.
- $(numOf(Engineer) \wedge numOf(ActivityPlanner) \wedge numOf(ActivityLeader) \wedge numOf(Student) \wedge numOf(ECAADPlanner) \wedge numOf(ECAADUI)) >= 1$
  Every role is filled at least once.

3. Architectural design

The next phase of the GAIA Method is the **architectural design**. In the analysis phase the main goal was to understand the system, without making final decision, but giving a flexible and extensible model. During the design phase we will extend the preliminary models until we decide that we determined the system's concrete attributes.

Relationships in the organization can be any type if their meaning is well defined in the collaboration. There are 3 fundamental types that are frequently used in several cases.

1. Control - When a given role has some ability to control other roles with certain protocols or manipulating environmental dependencies with activities that the controlled role heavily depends on. Representing a very strong authority relationship.

2. Peer - Peer roles has the same technical behavior in the environment, therefore a kind of collective sub-group of the organization, representing their equal status.

3. Depend - The inverse of the control relationship. Representing that the role heavily relies on another role's knowledge and resources.

For the ECAAD Engineering sub-organization the structure is really simple, as they handle the availability of OLM and ECAAD resources. However in the NEXT-TELL documentation we can read, the teachers may be connected to quality assessment phases later. Therefore we will include to the ECAAD Engineering sub-organization a QA and Modelling team.

Formal representation of the organization structure is the following:

1. **ECAAD Engineering sub-organization**
   - $\forall i, j : Engineer[i] \leftrightarrow^{peer} Engineer[j]$
   - $\forall i : Engineer[i] \rightarrow^{depends-on} ECAADQA$
   - $\forall i : ECAADQA \rightarrow^{depends-on} Engineer[i]$
2. **ECAAD Using sub-organization**
   - $ActivityPlanner \leftrightarrow^{depends-on} ECAADPlanner$
   - $ActivityLeader \leftrightarrow^{depends-on} ECAADUI$
   - $ECAADPlanner \rightarrow^{Control} ECAADUI$
   - $\forall i : Student[i] \rightarrow^{depends-on} ECAADUI$
   - $\forall i, j : Student[i] \leftrightarrow^{peer} Student[j]$

In order to make a clear view about the **interaction model**, one can define the rules as main scenarios of role communications. Such an example can be in the NEXT-TELL project when the teacher initiates a learning activity, or when the learning activity is about to end. These figures can be found in the study[4].

We gave the interaction model with a graphical representation so we assume that the roles satisfy the simulation needs of the ECAAD system behavior in a certain level. However we still need to define the **activity model** for our roles and extend their liveness and safety properties to satisfy organization structure attributes. The activities for the roles are the following:

- **Activities of the ECAAD Engineer**
  - ***MethodParticularisation***(Initiator: ECAAD Engineer)
    Resource: personal storage
    Description: The engineer particularize methods based on different aspects and target audiences.
    Output(Label: modelling): New method model
  - ***MethodImplementation***(Initiator: ECAAD Engineer)
    Resource: personal storage
    Description: The engineer implementing methods based on a particularization.
    Output(Label: implement): New method implementation

  - ***MethodDeployment***(Initiator: ECAAD Engineer)
    Environmental Resource: Assessment Library, Activity Library
    Description: The engineer deploys the modelled and implemented ECAAD model into the libraries.
    Output(Label: deploy): New method model

- **Activities of the ECAAD Planner**
  - ***PrepareActivity***(Initiator: ECAAD Planner)
    Environmental Resource: Activity Sequence
    Description: The tool is loading the learning activity and it's assessment method to it's scope and provide it for users.
    Output(Label: setup): Prepared Activity.
  - ***MonitorProgress***(Initiator: ECAAD Planner)
    Environmental Resource: Activity Sequence
    Description: The tool is continuously tracing and monitoring the activities inside the loaded learning activity sequence.
    Output(Label: trace activity, Note: monitor): Gathered information.
  - ***SaveResults***(Initiator: ECAAD Planner)
    Environmental Resource: Assesment Library, Activity Library
    Description: After the collected information of the monitored progress, the statistics are saved into the loaded model.
    Output(Label: update): Saved results into the activity sequence.

- **Activities of the ECAAD Student**
  - ***MakeProgressInActivity***(Initiator: ECAAD Student)
    Resource: ECAAD UI
    Description: The student tries to proceed through the currently loaded learning activity.
    Output(Label: using tool): Solved tasks.

4. Detailed design

The last step of the GAIA modeling is the **detailed design** declaration. The goal of this step is the identification of the **agent** and **services models**, that will stand as the final guideline for the actual implementation of the found agents and their activities.

4.1. Agent model

In the GAIA methodology agents stand as active software entities that are playing a set of agent roles. Usually if we are focusing on to keep the organization structure simple, or we defined a similar structure like the real world organization, we will found a high number of one-to-one correspondence between roles and agents.

In the current case study we define the following agents:

- **ECCAADEngineerAgent** - This agent plays the ECAAD Engineer role:
  $ECAADEngineerAgent \xrightarrow[1..N]{plays} ECAADEngineer$

- **ECCAADToolAgent** - This agent plays both the ECAAD Planner and UserInterface roles:
  $ECAADToolAgent \xrightarrow[1]{plays} ECAADPlanner, ECAADUI$

- **ECCAADTeacherAgent** - This agent plays both the ActivityPlanner and ActivityLeader roles:
  $ECAADTeacherAgent \xrightarrow[1]{plays} ActivitiyPlanner, ActivityLeader$

- **ECCAADStudentAgent** - This agent plays the ECAAD Student role:
  $ECAADStudentAgent \xrightarrow[1..M]{plays} ECAADStudent$

4.2. Services model

In the services model we need to identify so called services for every agent class. We need to keep in mind that services are not standard methods or functions, but a single and coherent piece of activity. For each activity one need to define it's attributes as follows.

- Inputs and outputs: These will be derived directly from the finalized protocols.

- Pre- and post-conditions: These are defined based on the safety properties of the roles.

The service definitions are found in table 6 and 7.

Table 6: The services model's inputs and outputs.

Service	**Inputs**	**Outputs**
activity planning	activityDescription	learningActivity
start activity	loadedActivity	startedActivity
monitor activity	runningActivity	
progress in activity	runningActivity	answersForTasks
finish activity	runningActivity	finishedActivity
get feedback	finishedActivity	feedback
save results	feedbacks $\wedge$ collectedInformation	

Table 7: The services model's pre- and post-conditions.

Service	Pre-condition	Post-condition
activity planning	$toolsAreUsable = true$	$actIsReady = true$
start activity	$toolsAreUsable = true$	$actIsStarted = true$
monitor activity	$activityIsStarted = true$	$true$
progress in activity	$activityIsStarted = true$	$feedbRequest$
finish activity	$numOf(finished) = MAX$	$monitorStop = true$
get feedback	$feedbackRequested = true$	$gotFeedb = true$
save results	$resultsAreSaved = true$	$modelIsUpdated = true$

5. Summary

In this case study we successfully gave a simplified agent oriented system, followed up and modelled with the GAIA methodology. We need to emphasize that this modeling may be precised with taking into account all NEXT-TELL elements. In this contribution we highlighted, how the ECAAD system's usage can be modelled with the GAIA methodology, therefore the virtual classroom was formed with the highest priority. Further research may be the modeling of missing elements or even the simulation of the existing GAIA based ECAAD system.

Acknowledgments. I would like to express my gratitude to László Varga, my supervisor for the help in this contribution and for the guide to the world of multi-agent systems, and GAIA modelling aspects.

References

[1] Zambonelli, F., Jennings, N. R., Wooldridge, M., (2003). Developing multi-agent systems: The Gaia methodology. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 12(3), 317-370.

[2] NEXT-TELL - Next Generation Teaching, Education and Learning for Life Project, (2014) *Information and Communication Technologies (ICT) theme of the 7th Framework Programme for R&D (FP7)*, http://www.next-tell.eu/

[3] Michael, W., Nicholas, R. J., David, K., (2000). A Methodology For Agent-Oriented Analysis And Design, *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3):285-312.

[4] Modelling the NEXT-TELL project with the GAIA methodology, (2014), https://rsipos.web.cern.ch/rsipos/GAIA-NextTell.pdf