INGENIAS-Scrum

Juan C. González-Moreno, Alma Gómez-Rodríguez, Rubén Fuentes-Fernández, and David Ramos-Valcárcel

Abstract
This chapter introduces the definition of an agile process for the INGENIAS methodology. It is based on a well-known development process: Scrum. The process adopts the iterative and fast plan presented originally by the methodology and uses some of the activities and most of the work products of the INGENIAS proposal with the Unified Development Process (UDP) (introduced in a previous chapter). The new approach is also based on the INGENIAS metamodel, but it is more focused on code development than on system specification. It takes advantage of the INGENIAS Agent Framework (IAF), which is part of the INGENIAS Development Kit (IDK). As this approach uses the same metamodels than the UDP based, there are not great differences in the models of the case study, but, instead, the organization of the work products and the time spent to get the final results is quite different.

1 Introduction

Agent-Oriented Software Engineering (AOSE) methodologies have adopted different development processes depending on their particular needs and what the prevalent processes in mainstream software engineering [1] were. From [2], the process models used in AOSE can be classified into the following groups:

- *Waterfall*. The waterfall-like process models prescribe a sequential, linear flow among phases. Although some waterfall models include some kind of return to

J.C. González-Moreno • A. Gómez-Rodríguez (✉) • D. Ramos-Valcárcel
Universidad de Vigo, Campus As Lagoas s/n, 32004 Ourense, Spain
e-mail: alma@uvigo.es; jcmoreno@uvigo.es; david@uvigo.es

R. Fuentes-Fernández
Universidad Complutense de Madrid, Avda. Complutense s/n, 28040 Madrid, Spain
e-mail: ruben@fdi.ucm.es

previous phases, in real projects this return occurs very late and the cost of any change in the initial specifications is unacceptable.

- *Evolutionary and Incremental*. The stages of this process model category consist of expanding increments of an operational software product, with the direction of evolution being determined by operational experience. Most of AOSE methodologies have process models within this category.
- *Transformation*. This software development may be seen as a sequence of steps that gradually transform a set of formal specifications into an implementation.
- *Spiral*. It organizes the development process in a cyclic way. Each cycle of the spiral consists of four phases: determining the objectives, evaluating the risks of these objectives, developing and verifying, and reviewing the previous stages.

Among all of them, agile processes [3] can be included in *Evolutionary and Incremental* class. This means that the development is organized to deliver functional increments of high value for the customer in short periods of time. These processes are code-oriented in the sense that they consider the main product of development is functional code.

Scrum [4] is an empirical agile project management framework. It relies on self-organizing, empowered teams to deliver the product increments, but also on a customer, or *Product Owner*, that must provide the development team with a list of desired features, using business value as the priority mechanism. In real life, *Scrum* is a mechanism in the sport of rugby for getting an out-of-play ball back into play. The term was adopted in 1987 by Ikujiro Nonaka and Hirotaka Takeuchi to describe *hyper-productive development*. Jeff Sutherland developed the Scrum process in 1993 while working at the Easel Corporation, and Ken Schwaber formalized the process in the first published paper on Scrum at OOPSLA 1995 [5].

The Scrum process framework is specially well suited for Knowledge Engineering Developments based on the use of multi-agent systems (MAS). This chapter addresses its integration with the INGENIAS methodology [6] using some specific tools. INGENIAS is a general purpose AOSE methodology that adopts a model-driven development (MDD) [7]. Its development is organized around the definition of models that are semi-automatically transformed into different products, for example, code, tests, and documentation.

The INGENIAS modeling language is defined through a metamodel, following common practices in MDD [8]. It is the basis to define the transformations that generate the code of support tools and MAS in INGENIAS. This metamodel has proved its capability and maturity in the development of MAS for different domains [6, 9, 10]. The main support tool is the INGENIAS Development Kit (IDK) [11], which includes a graphical editor for MAS specifications compliant with the INGENIAS metamodel, and integrates plug-ins to provide additional functionalities. Among these plug-ins, those to manage developments using the IAF are the most relevant ones.

The IAF [12] is a set of libraries that facilitate the implementation of INGENIAS agents built on top of the Java Agent DEvelopment Framework (JADE) [13]. The tool has been proposed from the experience in the application of the INGENIAS methodology over several years to enable a MDD.

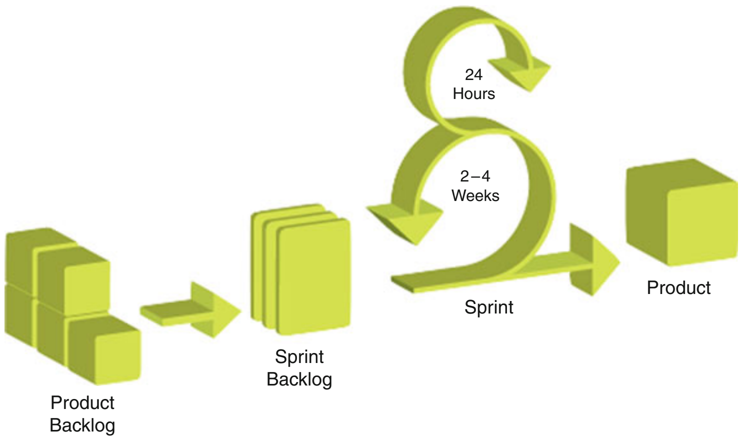


Fig. 1 Scrum life cycle

The IAF is fully integrated in the IDK [11] and provides facilities for project checking, code generation, debugging, and documentation. This means that following the guidelines of the IAF documentation, an experienced developer can focus most of its effort on specifying the system, converting a great deal of the implementation in a matter of transforming automatically specifications into code. This quick transition from specifications to code facilitates the adoption of agile processes such as Scrum [4].

The rest of this chapter fully specifies the development process based on the Scrum framework and the use of the IAF proposed in [1]. The *INGENIAS with the UDP chapter* in this book also considers the INGENIAS methodology, but with its original process based on the Unified Development Process (UDP) [14]. The interested reader can find additional discussion on the INGENIAS methodology in that chapter.

1.1 The INGENIAS-Scrum Process Life Cycle

Scrum [4] is a suitable process for contexts where developers consider short and incremental development cycles focused on running code. This scenario is enabled by the IAF [12] for the INGENIAS methodology (Fig. 1).

Scrum usually addresses the production of a software version (i.e., a *release*) every couple of months. Potential features to accomplish in a release are collected in the *product backlog* and prioritized. A *product owner*, who represents the customer, is in charge of updating this backlog. The release is produced in a number of iterations called *sprints*. Each sprint usually lasts from 2 to 4 weeks. The sprint content is defined by the *product owner* taking into consideration both priorities and team capabilities. The team defines the tasks required to develop the functionalities

Preparation Phase
Sprint Phases

Fig. 2 Scrum phases

selected for the sprint. Within a sprint, progress checkpoints are performed during the *daily scrums*. This enables the *scrum master* to assess work progress regarding the sprint goals, and to suggest adjustments to ensure the sprint success. At the end of each sprint, the team produces a potentially releasable *product increment*, whose evaluation drives the *backlog update* for the next sprint.

All this work is planned in two kinds of phases (see Fig. 2): the *Preparation* phase and the *Sprint phases*. The Preparation phase includes all the activities to be done before the first sprint, focused on the elaboration of the product backlog. The Sprint phases covers the execution of all the sprints required to release the final product. The following sections discuss the key issues of the integration of this process with INGENIAS through model-driven practices.

1.2 Metamodel

The INGENIAS metamodel describes the elements that constitute a MAS according to this methodology. This metamodel is explained in detail in the *INGENIAS with the UDP chapter*, so this chapter only offers a short overview. A detailed discussion of the metamodel can be found in [15, 16].

This metamodel considers several viewpoints (i.e., *models*) that correspond to different aspects of a MAS. Figure 3 (shared with the *INGENIAS with the UDP chapter*) shows them and their relationships. The viewpoints are

- The MAS organization, that is, the *Organization Model*.
- The definition, control and management of each agent mental state, that is, the *Agent Model*.
- The tasks and goals assigned to each agent, that is, the *Tasks and Goals Model*.
- The agent interactions, that is, the *Interaction Model*.
- The external environment where agents interact to satisfy their goals, that is, the *Environment Model*.

1.3 Guidelines and Techniques

The process proposed for INGENIAS [6] must be able to guide developers to obtain the products that meet the development goals. Being INGENIAS a MDD-oriented methodology, this means that once a specification conforming to the metamodels is defined, the products are generated through transformations and using, if needed, additional resources (e.g., code templates, external code, or scripts).

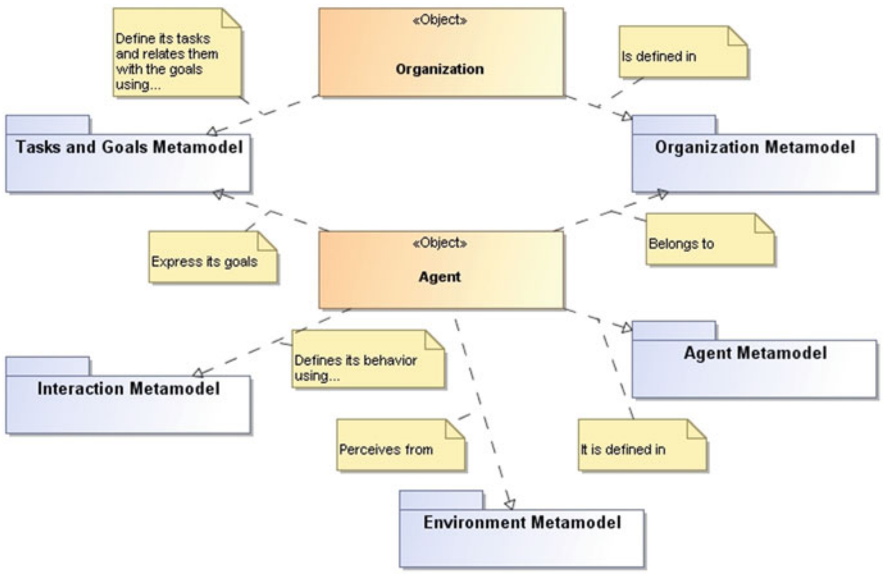


Fig. 3 Metamodel of the INGENIAS methodology

1.3.1 Scrum and MDD

Although Scrum [4] does not describe the engineering activities required for product development, the application of Scrum to INGENIAS does it. INGENIAS-Scrum relies on the use of the IAF [12], which allows to combine the classic approach for coding applications with the modern techniques of automatic code generation.

An IAF compliant specification must describe in a sound way the following aspects:

- *Agents*. It is the main building block in an INGENIAS MAS. An agent is defined completely when its perception, main tasks, and coordination means are described.
- *Tasks*. They are the basic units of agent's behavior. A task modifies the agent mental state and performs actions over the applications.
- *Interactions*. They describe how agents coordinate in order to satisfy their goals. Coordination is defined in terms of transferred information units between agents.
- *Deployment*. It expresses how agents are instantiated and initialized in the final system.
- *Tests*. The test define the testing units that the MAS should pass through to ensure the system will achieve its goals and will work in a sound way along its life cycle.

This information is provided through several INGENIAS models (see Sect. 1.2). The core ones, in this case, are the *Agent Model*, the *Interaction Model*, and the *Tasks and Goals Model*. The *Environment Model* may also be considered, mainly to point out the relationships with external elements. Nevertheless, its elements can be defined by using the rest of models, particularly using the Agent Model (as shown in Fig. 3). Regarding the *Organization Model*, it is not strictly necessary to use it, but it may help to have a global vision of the final solution in what refers to *agents and roles that participate*, and *responsibilities in the satisfaction of the main goals*. Moreover, it may be used to reflect many of the product items proposed in the original Scrum framework.

Note that although not all the INGENIAS models are strictly necessary to create an IAF compliant specification, this does not mean that the models do not provide information usable for code generation. For instance, the concepts of *Autonomous Entity* and *Organization* presented in *Table 2 of the INGENIAS-UDP chapter* could have a translation to code in the form of new capabilities or relationships.

1.3.2 IDK Considerations

The IDK [6, 11] is a graphical tool for MAS model creation. Although it is fully adapted to cover the INGENIAS metamodel, the editor offers several features that allow the adaptation to new metamodels or target platforms. At present many of such features could be used to accurately support the documentation provided for some non-standard artifacts like *user stories*, *cards*, *priorities*, *estimations*, etc. For instance, a good practice to cover users stories is to associate a package to each one and use the text description to introduce a full description of the story. This text description may be used also to include information about priority, velocity, tasks assignation, etc. After, this description can be refined and formalized by using the diagrams of the metamodel that is associated with such package. Another good complementary practice is to use a naming protocol for each package to document the version and level of development for such package. The use of that package naming convention also facilitates the implementation of an iterative and incremental process for any system. The capability of the tool, which allows to move diagrams from a package to another one, is also fundamental to facilitate such kind of process.

1.3.3 IAF Considerations

During the definition of the product backlog using the INGENIAS models, participants must take into account that an agent in the IAF performs a simple deliberation cycle:

- Identify new tasks to schedule and tasks to remove from the schedule
- Execute one scheduled task

This cycle omits the classic perception step as it is not needed. In fact, the agent can receive new information at any moment and this does not cause internal conflicts. The incorporation of new pieces of information does not imply a change in already scheduled tasks, though it may mean the incorporation of new tasks.

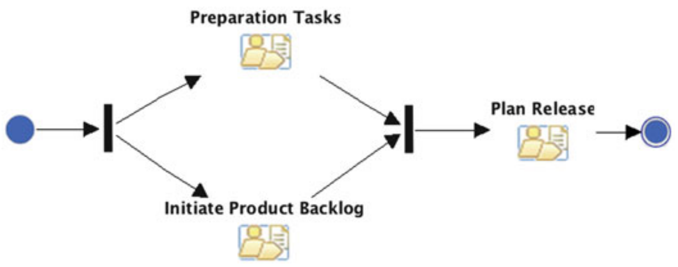


Fig. 4 The Preparation phase flow of activities

2 Phases of the INGENIAS-Scrum Process

2.1 Process Documentation: Preparation Phase

As the Scrum framework does not take into account all the aspects of project planning, the main goal of the Preparation phase is to define the *Plan Release*. This implies establishing the features to address in the release and the estimated tasks to achieve them.

Accordingly with the original Scrum framework, the *Preparation* phase comprises the following activities: *Initiate Product Backlog*, *Plan Release*, and *Preparation Tasks*. Figure 4 shows the workflow of these activities, and Fig. 5 shows the structure of the phase in terms of the tasks and work products to be accomplished. Section 2.1.1 provides information about the roles responsible of each task and the kind of responsibility they assume. Section 2.1.2 details the activities and their tasks, and a description of the work products related with those tasks appears in Sect. 2.1.3

It is interesting to note that several tasks of this phase have to be done using the IAF integrated in the IDK. The main guideline to apply correctly both INGENIAS tools is the technical report [12], which comes with the IDK distribution.

2.1.1 Process Roles

Following the Scrum approach, the roles implied in the Preparation phase are the *Product Owner*, the *Scrum Team*, and the *Stakeholder*. The following sections describe them in detail.

Product Owner

This role represents the *customer* during the development and is in charge of the *Initiate Product Backlog* activity. This activity produces a *backlog* containing enough features to allow the startup of a number of sprints. The role also participates in the *Plan Release* activity to set up the *release*'s initial planning in order to launch the sprints.

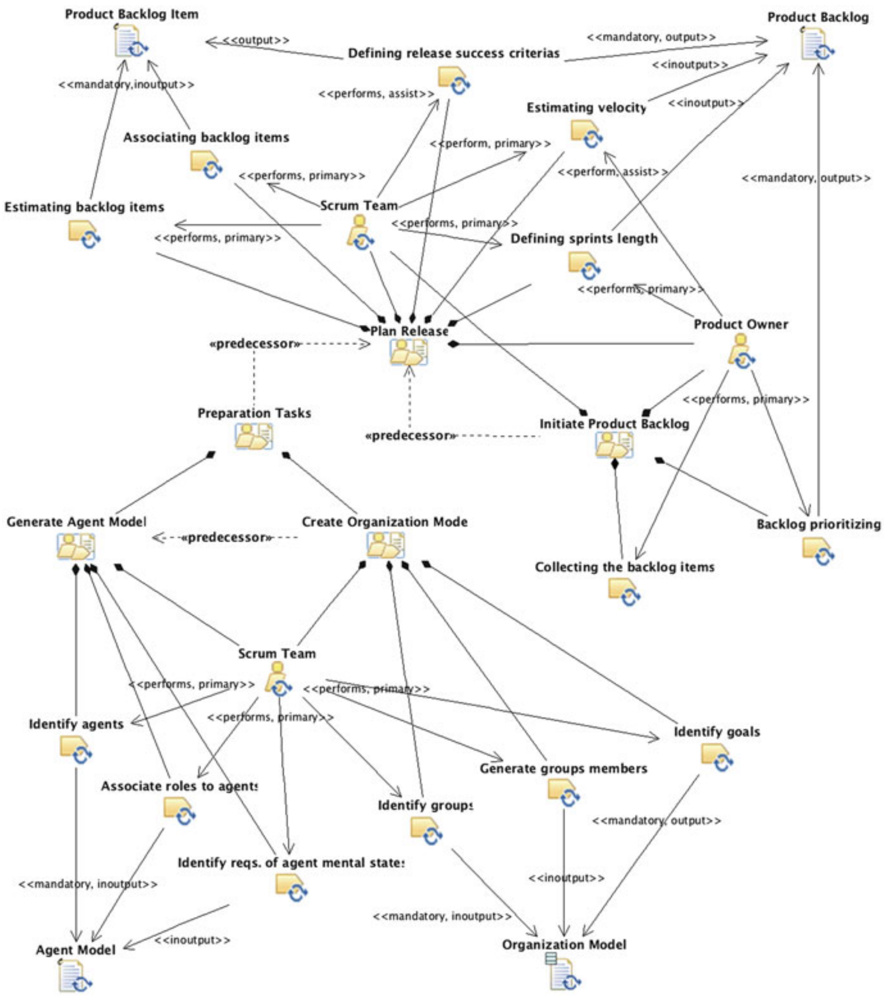


Fig. 5 The Preparation phase described in terms of activities and work products

To obtain the *backlog*, the Product Owner identifies a list of items, prioritizes them, and includes them in the *Product Backlog*. In a traditional approach, these items (on the preparation phase) consist of *User Stories* that details the requirements of the system. In the INGENIAS-Scrum approach, the Product Backlog will consist of an Agent Model and Organization Model (optionally). User stories are documented using the IDK by the Scrum Team and are stored on the description part (see Fig. 10) of the goals identified following the Product Owner indications. As this role is in charge of defining the functionality of the application, it is desirable that she/he could also determine the acceptance tests while prioritizing each item.

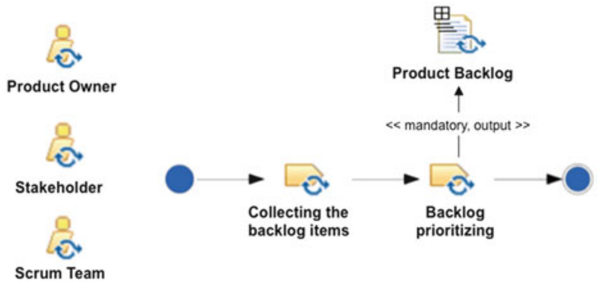


Fig. 6 Workflow of the *Initiate Product Backlog* activity

Scrum Team

It is mainly responsible of the *Preparation Tasks* activity that will be done jointly with *Initiate Product Backlog*. This role also participates actively in the rest of the activities of the *Preparation* phase. The main goal at this point of the life cycle is to specify an initial view of the system to be developed using the IDK and the INGENIAS metamodels (see Fig. 3). This view comprises the development of an *Organization Model* (optionally) and/or an initial *Agent Model* (mandatory) with several *Agent Diagrams*, including the *agents*, *roles*, and *goals* required by the Product Owner for the MAS. This initial view must include enough information to provide a planning for the first 2–3 sprints.

Stakeholder

The participation of this role is optional and represents, as usual, anyone that does not directly participate in the project but can influence the product under development.

2.1.2 Activity Details

Initiate Product Backlog

The product backlog is a prioritized list of features that contains short descriptions of all the functionality desired for the product. For INGENIAS, these requirements are described using the IDK, either by adapting a specification from a previous project or by defining a new one.

When using Scrum, it is not necessary to start a project with a lengthy, upfront effort to document all the requirements. Typically, a Scrum Team and its Product Owner begin by writing down everything they can think of easily, and this constitutes the first version of the product backlog. This is almost always more than enough for a first sprint (when documented accordingly with the IDK). The product backlog (i.e., the INGENIAS specification) is then allowed to grow and change as more information is learned about the product and the customers. The flow of tasks needed to perform this activity is described in Fig. 6 and detailed in Table 1.

Table 1 Description of the tasks of the *Initiate Product Backlog* activity

Activity	Task	Description	Involved roles
Initiate *Product Backlog*	Collecting the backlog items	The *Product Owner* establishes a first list of requirements (a requirement becomes a *Product Backlog Item*) that she/he wish to be implemented for the end of the release. After this, a workshop is organized in which the Product Owner participates, as well as, the whole team and the stakeholders (if necessary). The Product Owner introduces the backlog draft and the assistants may suggest additional items.	Product Owner Scrum Team Stakeholder
	Backlog prioritizing	The full list of backlog items is prioritized by the Product Owner. Items with higher priority must be addressed first in the release. If a backlog contains a lot of items, the definition of *Themes*, and the association of one of them to each item, is suggested. After this, the Product Owner prioritizes the *themes*.	Product Owner Scrum Team Stakeholder

Preparation Tasks

This activity is performed by the Scrum Team in order to prepare the next meetings with the Product Owner and the Stakeholders. It creates an ongoing INGENIAS specification intended to reorganize, formalize, and structure the preliminary specifications in the product backlog. This implies that this activity is more complex than the original Scrum one. It requires to perform several tasks which are common to the INGENIAS activities *Create the Organization Model* and *Create the Agent Model* that have been detailed on the *INGENIAS-UDP chapter*. Figure 7 shows the recommended flow of such tasks. As it can be easily seen, they differ a little from the ones proposed on the *INGENIAS-UDP chapter*.

At this stage of the process, the considerations introduced at Sect. 1.3.3 must be taken into account for creating and modifying the *Organization* and *Agent Models*.

Plan Release

The previous activities deliver an initial INGENIAS specification. With this information, the *Scrum Master*, jointly with the Product Owner, can establish the *Plan Release*, which defines the sprints required to meet the goals. Several tasks have to be performed, including the definition of the release success criteria, the estimation of the backlog items and the velocity of the sprint, the definition of the sprints length, and the association of the backlog items. Figure 8 shows the tasks workflow for this activity that Table 2 further explains.

2.1.3 Work Products

Work Product Kinds

The unique product produced by the Preparation phase is the initial *Product Backlog*. A Product Backlog is a set of items: the Product Backlog Items (PBI). A PBI in the INGENIAS-Scrum approach is a composed work product of type

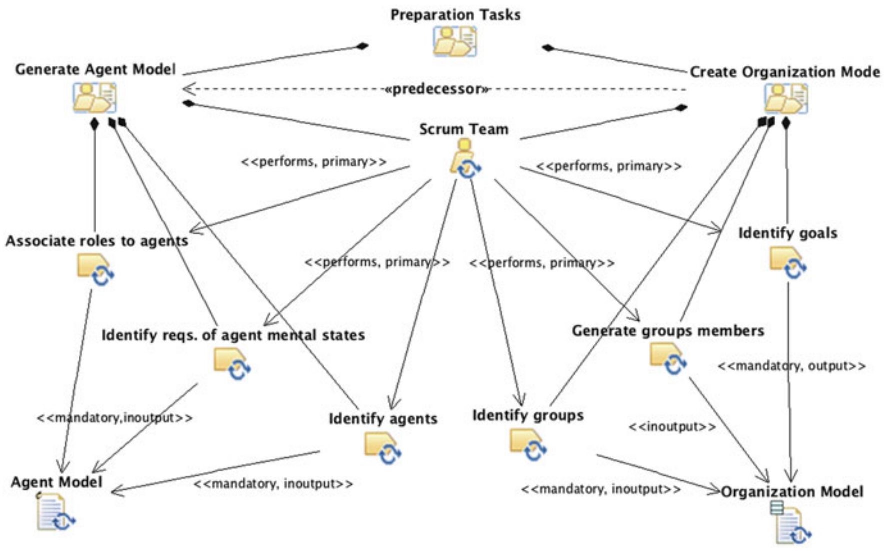


Fig. 7 Workflow of the *Preparation Tasks* activity

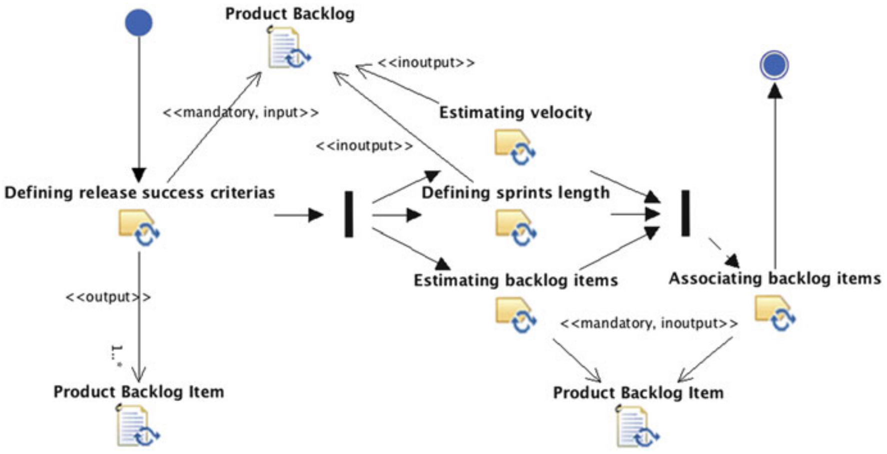


Fig. 8 Workflow of the *Plan Release* activity

Composite (Structured + Structural + Behavioural). In the original proposal, each PBI can be described as free text and refers to one of the following kind of items:

- *Features*. A feature usually is described with *User Stories* that comprise a short and simple description of the desired functionality from the user's perspective. An example would be, "As an author, I can send a paper to the conference and I will be informed about its reception and the identification number assigned to it"

Table 2 Description of the tasks to be done during the *Plan Release* activity

Activity	Task	Description	Involved roles
Plan Release	Defining *release success criteria*	This task pursues to establish the criteria to consider a Release successfully finished by the Product Owner. The Scrum Team (the Scrum Master, in particular) must take into account that there are 2 types of releases: *End-Date Driven Release*, which must be available to the end users before a deadline; and *Feature Driven Release*, in which the release finishes when all of the requirements are implemented.	Product Owner Scrum Team
	Estimating *backlog items*	Estimation must be performed by the team item per item. It is a good idea to start by the highest priority items. In this activity, it is usual to use as range of values for estimations the *Fibonacci Numbers*: 1, 2, 3, 5, 8, and 13. Regarding the *estimation techniques*, the collective ones are preferred.	Scrum Team
	Defining *Sprint length*	Although historically the length of a sprint is 30 days, INGENIAS-Scrum recommends 15 days, although a week or 21 days are also acceptable lengths, according to the difficulty of the work and the human resources available.	Scrum Team
	Estimating *velocity*	The *velocity* (i.e., the number of items that could be finished) is calculated as the sum of all items checked and approved as fully implemented in a Sprint. This activity makes an estimation of the expected team velocity. It should be done automatically because it is supposed that the team has worked together in previous projects using the INGENIAS tools and the JADE platform. Nevertheless, the velocity can be manually estimated for the first time and adjusted in the next sprints.	Scrum Team
	Associating *backlog items* to *sprints*	This task takes into consideration the parameters obtained from the previous tasks. A slight adjustment of items prioritization may be done at this stage, so the velocity can be better adjusted to the team features. It is not required to make this association for all sprints of the release at the beginning, but it should be done for the 2 or 3 first ones.	Product Owner Scrum Team

- *Bugs*. In general, a bug does not present any real difference with a new feature (in the description), and it will be treated as if a new feature or a change is to be incorporated.

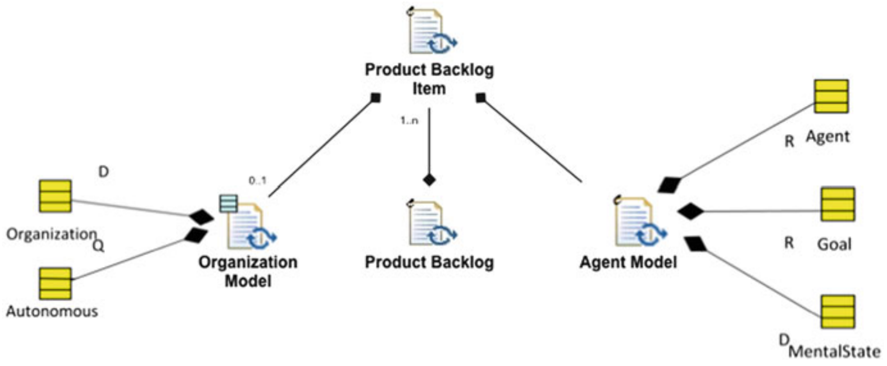


Fig. 9 The Preparation Work Product Model structure

- *Technical work*. This kind of item introduces detailed aspects related with the deployment of the system. An example of technical work would be: "Upgrade the Conference Server Operating System to OS X Mountain Lion" or "Use a version up to 2.0 of the Mail Manager system".
- *Knowledge acquisition*. This item is related to obtain the skills and knowledge required for the project. The item reflects some kind of external knowledge that is necessary to better understand the problem or to apply a particular solution. An example could be asking the team members the study and/or the selection among several Java libraries to determine the most accurate one for the system or to find a solution to the problem restricted to the use of such library.

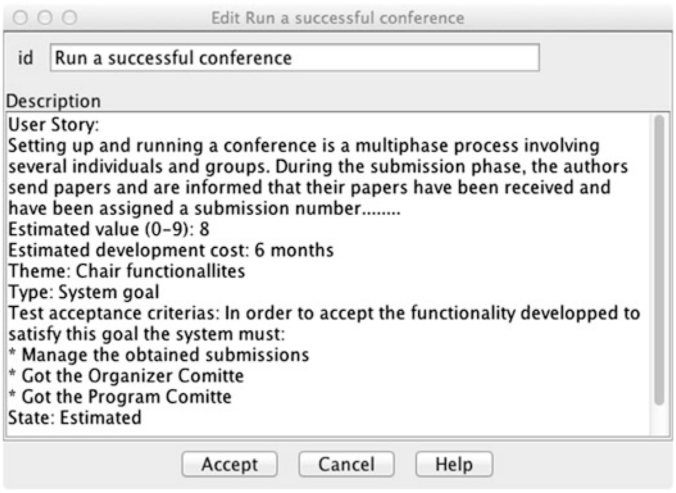
User stories usually determine several goals that must be satisfied by agents playing certain roles. Bugs must be documented by modifying the description of the related user story or adding a new one. Regarding technical works and knowledge acquisition items, a good practice is to document them by defining a test or proof for a part of the system, or by specifying the concrete API needed to solve the problem jointly with the agent that is in charge of applying such solution.

In INGENIAS-Scrum, a Product Backlog is described by a set of INGENIAS models as reflects Fig. 9. For a full description about the *Organization* and *Agent Model*, see *INGENIAS-UDP chapter*. At the Preparation Phase, each PBI, as described by the Product Owner and the Stakeholders, is specified as a Package description as pointed in Sect. 1.3.2, which is refined by the Scrum Team by using agent and/or organization diagrams to determine the Agent and/or Organization Models. Agent Model is mandatory at this phase, but could be complemented with an Organization Model to provide the responsibilities of groups, agents, and roles in order to satisfy the identified goals. Figure 9 shows the relationships between the MAS meta-model elements.

Product Backlog Item

In general, a PBI has to include the following general attributes:

- Estimated value, frequently calculated relatively to other items using priorities.
- Estimated development cost.



id: Run a successful conference

Description

User Story:
Setting up and running a conference is a multiphase process involving several individuals and groups. During the submission phase, the authors send papers and are informed that their papers have been received and have been assigned a submission number........
Estimated value (0–9): 8
Estimated development cost: 6 months
Theme: Chair functionallites
Type: System goal
Test acceptance criterias: In order to accept the functionality developped to satisfy this goal the system must:
* Manage the obtained submissions
* Got the Organizer Comitte
* Got the Program Comitte
State: Estimated

Fig. 10 Attributes of a Product Backlog Item specified in the description of an INGENIAS element

- Eventually the *theme* (i.e., functional domain) it belongs to.
- Eventually its type, which can be defined as: *system goal of the project*; *bug resulting from the process*; or *non-functional requirement*.
- The associated criteria for test acceptance.

Through this phase of the development process, a PBI can reach the following states: *created*, *estimated*, *planned* (associated to a future sprint), *associated to current sprint* (implementation is ongoing), and *done*.

At the end of the Preparation Phase, following the INGENIAS-SCRUM approach, all the PBI identified jointly with their state must have at least one *Goal* associated and its features must be documented in the description of the related package and detailed on the related goal description. Figure 10 shows a snapshot of how this information could be documented using the IDK.

2.2 Process Documentation: The Sprint Phases

The *Sprint* is the main phase of a Scrum project. The INGENIAS-Scrum process is iterative and incremental, which means that the project is split into consecutive sprints that will be done with the help of the IDK and IAF tools. In the original Scrum framework, each sprint is timeboxed between 1 week and a calendar month. In the INGENIAS-Scrum approach, no more than 21 days for a sprint are suggested because the most common sprint length for the Scrum framework is 2 weeks and the use of the INGENIAS model-driven tools promotes short cycles. The usual goal of each sprints is to implement and test JACE code covering at least one of the MAS goals. Figure 11 shows the workflow of the activities to be accomplished during the Sprint Phase.

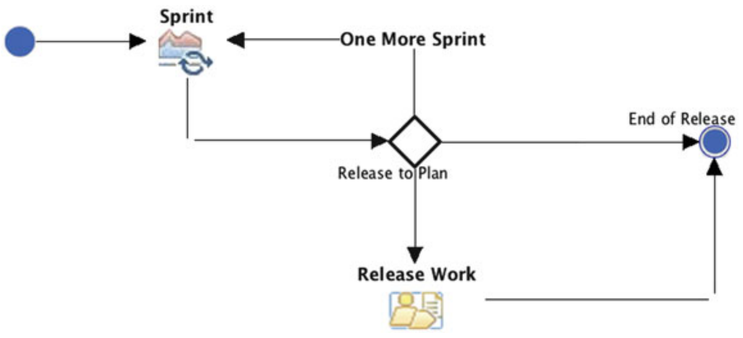


Fig. 11 The Sprint Phases flow of activities

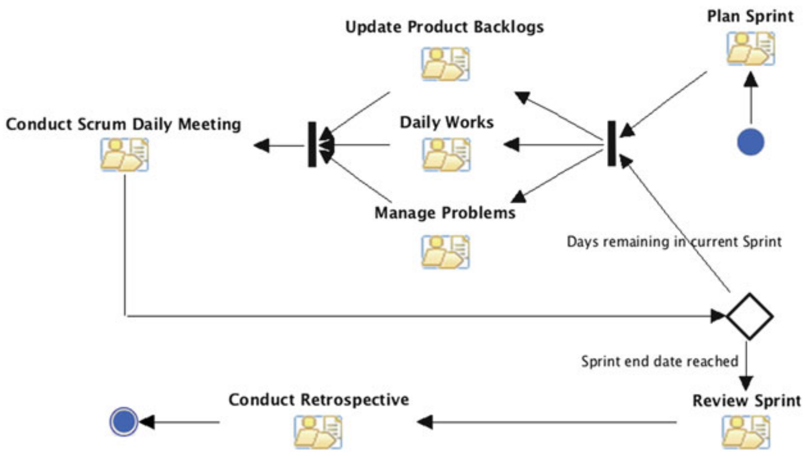


Fig. 12 The Sprint activity workflow

Current Phase (see Fig. 11) is composed of an iterative and complex phase, the proper *Sprint* and an optional activity the *Release Work*. The Sprint phase is composed of several activities whose workflows are shown in Fig. 12 and that is described in detail in Sect. 2.2.2. Section 2.2.1 introduces the roles involved on each Sprint. Finally, Sect. 2.2.3 details the work products of the *Sprint Phases*.

2.2.1 Process Roles

Following the Scrum approach, the roles that are implied in the Sprint Phases are described in the following sections.

The Product Owner

During each Sprint Phase, the Product Owner is responsible of analyzing the suggested changes in order to add them to the backlog and prioritize the added items. This is done before the next *Sprint Planning*. Ideally, this work is done 1 or 2 days before the *Sprint Review* during a brainstorming session with the Scrum Team.

The Master Team

The iterative Sprint Phases perform the product development. The *Master Team* is the responsible for helping the Scrum Team to work autonomously and constantly improving itself by doing the following kinds of tasks:

- *Periodical tasks*, whose main goal is to organize and promote the collaborative work during the meetings: *Daily Scrum*, *Sprint planning*, *Sprint review*, and *Retrospective*.
- *Event tasks*, whose objective is to remove impediments. This task mainly relies on taking into consideration previous events to solve recurrent problems as soon as possible, while protecting the team from external distractions.
- *Background task*, in which the Master Team tries to ensure that the team remains productive and focused on the project goal: developing backlog items in close collaboration with the Product Owner.

The Scrum Team

In the Scrum framework, the teams are composed of 3–10 members. In the INGENIAS approach, this number should be between 3 and 5 people. The *Scrum Team* is the main responsible of the *Product Increment* and its participation is key in the activities: *Plan Sprint*, *Update Product Backlog*, *Daily Works*, and *Conduct Scrum Daily Meetings*. Also, its participation is important on the *Review Sprint* and *Conduct Retrospective* activity. While it performs a secondary role at the *Manage Problems* activity. By participating in these activities, it modifies the *Product Backlog* and the *Sprint Backlog*.

The Stakeholder

Like in the *Preparation* Phase, the participation of *Stakeholders* is not necessary in any of the Sprint Phases. They can participate playing a secondary role in the *Update Product Backlog*, *Conduct Retrospective*, and *Review Sprint* activities. Their participation is consultive, they make suggestions about the goals that the system must satisfy and the right solutions from the customer perspective.

2.2.2 Activity Details

Figure 12 shows the workflow of the activities to be performed at each Sprint. The full structure of this Phase in terms of activities, tasks, and workproducts is presented in four figures: Figs. 13, 14, 15, and 16.

Figure 13 shows the structure of the activities *Plan Sprint* and *Update Product Backlog*. The *Plan Sprint* is the first activity to be performed on each Sprint, while the *Update Product Backlog* is done in parallel with the *Daily Works* and the *Manage Problems* activities.

The structure presented in Fig. 14 refers to the most complex activity of the full process. It represents the *Daily Work* of the Scrum Team that must be accomplished using the IAF with the IDK tool.

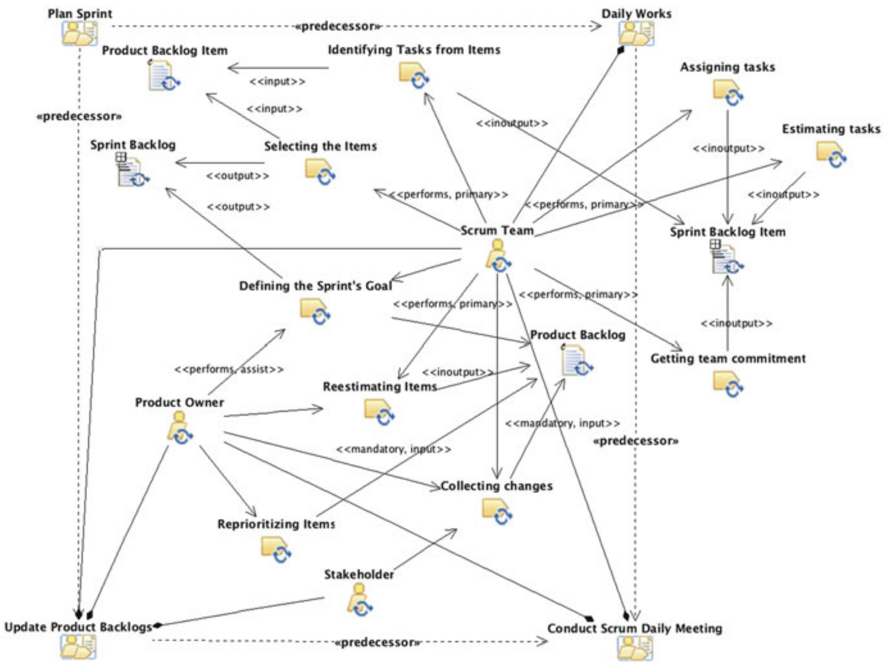


Fig. 13 Partial structure of the first activities done on each Sprint Phase

Figures 15 and 16 show the structure and the dependencies among the activities that control and determine the evolution of each sprint during the current release.

In the following subsections, all the activities jointly with their tasks are detailed and the workflow suggested to each one are showed.

Plan Sprint

The first activity of each sprint is to perform a *Sprint Planning Meeting*. During this meeting, the *Product Owner* and the *Scrum Team* discuss about the highest-priority items in the *Product Backlog*. External stakeholders may attend by invitation, although this is unusual in most cases. Team members determine the number of items they can commit to accomplish and then they create a sprint backlog, which is a list of the tasks to perform during the sprint. Figure 17 shows the workflow of the tasks with the work products and roles involved in this activity, and Table 3 describes in detail the tasks.

Update Product Backlog

The main goal of this activity is to update the backlog and adjust the planning, taking into account changes emerged since the last sprint. During a sprint, no one can modify the selection of backlog items done at the beginning of the sprint, that is, the sprint scope remains unchanged. However, anyone, even a external Stakeholder,

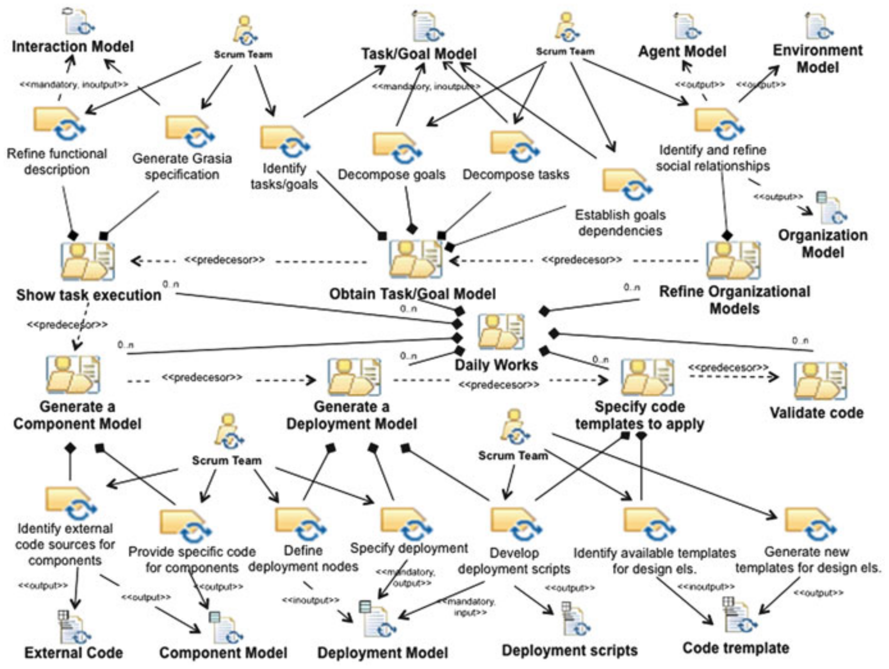

Interaction Model
Scrum Team
Task/Goal Model
Agent Model
Environment Model
Refine functional description
Generate Grasia specification
Identify tasks/goals
Decompose goals
Decompose tasks
Identify and refine social relationships
Establish goals dependencies
Organization Model
Show task execution
Obtain Task/Goal Model
Refine Organizational Models
Daily Works
Generate a Component Model
Generate a Deployment Model
Specify code templates to apply
Validate code
Identify external code sources for components
Provide specific code for components
Define deployment nodes
Specify deployment
Develop deployment scripts
Identify available templates for design els.
Generate new templates for design els.
External Code
Component Model
Deployment Model
Deployment scripts
Code trepmlate

Fig. 14 Structure of the *Daily Work* activity on each Sprint Phase

can suggest new items for later sprints. These items are studied and prioritized by the Product Owner prior to the next sprint planning. Bugs and enhancement requests, coming from *partial product tests* done at the end of the previous sprint, can also be used to update the backlog. Figure 18 shows the relations between tasks, work products, and roles in this activity, and Table 4 describes its tasks in detail.

Daily Works

The team performs backlog tasks to reach the sprint goals. The original Scrum templates provide no information for technical design, coding, and test tasks. Tasks are not assigned by the Scrum Master, but chosen by the team members one each time. Team updates (when necessary) the estimation of their remaining work to do in the Sprint. The recommendation of the INGENIAS-Scrum process for this activity is to merge and distribute, accordingly to the needs, the tasks presented in the Elaboration and Construction subsections of the *INGENIAS-UDP chapter* for the following activities:

- Refine organizational models with social relationships, which includes refining the Organization, Agent, and Environment Models
- Create the Tasks and Goals Model
- Show tasks execution using the Interaction Model
- Create a Component Model

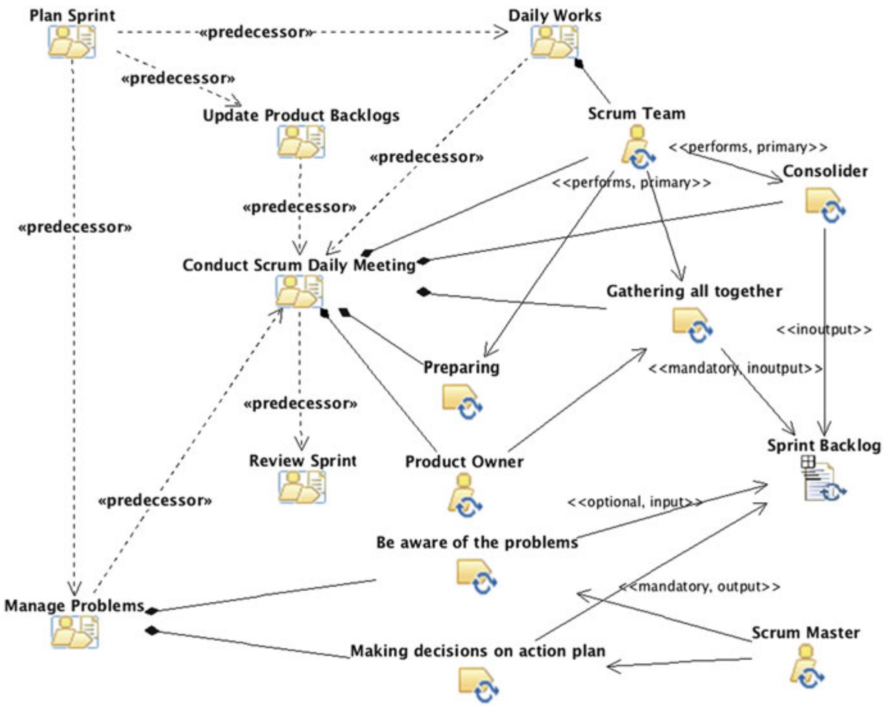


Fig. 15 Structure of the *Manage Problems* and *Conduct Scrum Daily Meeting* activities

- Create a Deployment Model
- Specify code templates to apply
- Validate code

Figure 14 shows the full structure of this activity, while the workflow and the details to accomplish such tasks are shown in the *INGENIAS-UDP chapter*.

Manage Problems

The Scrum Master takes into account the events that happen at any moment in a project. She/he tries to eliminate the problems experienced by the team members, so they can focus on their actual goals. The activity generates as output a Sprint Backlog and can use optionally as input a previous Sprint Backlog, as shown in Fig. 19. Table 5 describes the tasks to be done by the Scrum Master.

Conduct Scrum Daily Meeting

Every day of the Scrum sprint, all team members attend a *Scrum daily meeting*, including the Scrum Master and the Product Owner. This meeting is timeboxed to *no more* than 15 min. During the meeting, people participating in the development share their finished work, what they have to do that day, and identify any impediments or problems that could affect the team progress. This activity allows to synchronize

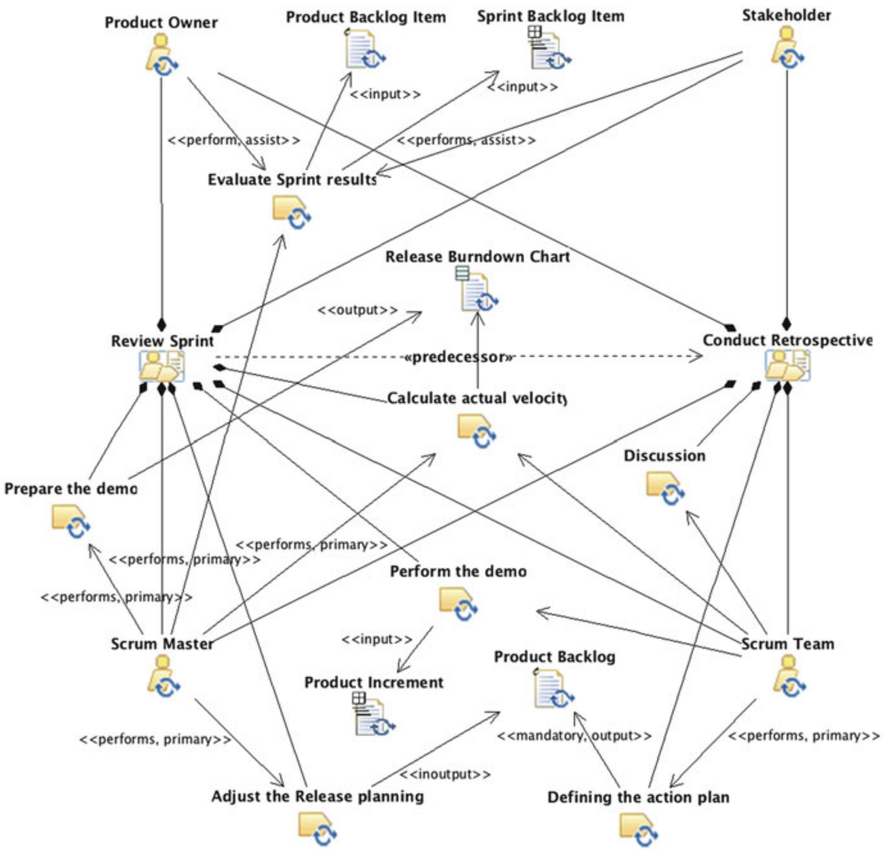


Fig. 16 Structure of the *Review Sprints* and *Conduct Retrospective* activities

ongoing work while discussing the progress of the current sprint. Figure 20 shows the relation between tasks, work products, and roles in this activity, and Table 6 details its tasks.

Review Sprint

At the end of a Scrum sprint, the team conducts a *Sprint Review* meeting. There, it demonstrates the functionality added during the current sprint using a demo. The main objective of this activity is to obtain feedback from the users invited to the review (such as the product owner or the stakeholders). The feedback can result in the acceptance of the work, the suggestion of changes to the delivered functionality, or even the revision or addition of backlog items. Figure 21 shows the relation between tasks, work products, and roles in this activity, and Table 7 details the tasks.

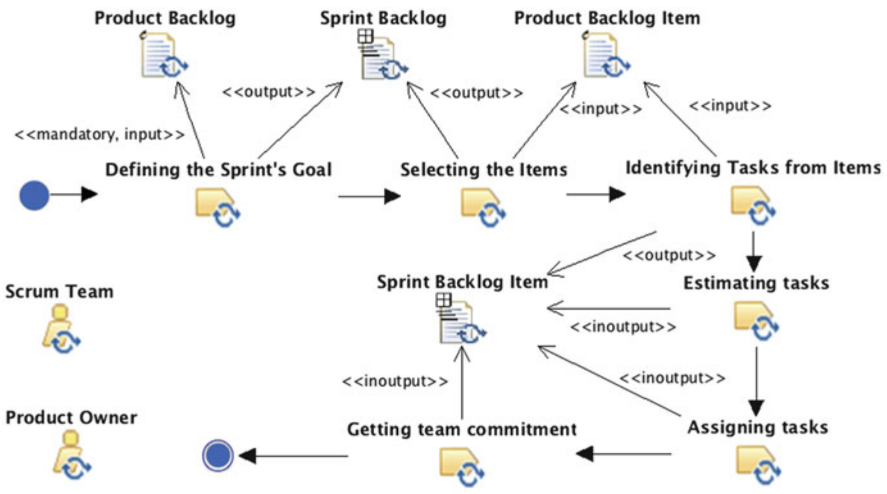


Fig. 17 Workflow of the *Plan Sprint* activity

Table 3 Description of the tasks of the *Sprint Planning* activity

Activity	Task	Description	Involved roles
Plan Sprint	Defining the *Sprint's Goal*	The goal of a sprint is proposed by the Product Owner. In the first sprints (2–3), it focuses on showing the feasibility of the potential architecture (i.e., the Organization and Agent Models). After architecture validation, the goals of sprints consist on satisfying a system *Goal*.	Product Owner Scrum Team
	Selecting the items	This task defines the scope of the sprint. The Scrum Team associates Product Backlog Items to the sprint. The team does it item by item, trying to balance the required effort with the team velocity. Then, the team collectively validates the subset of the backlog for the sprint.	Scrum Team
	Identifying tasks from items	This task is addressed in the second part of the meeting. Here, the team decides how to achieve the sprint goals. Each selected *Item* is decomposed into tasks to enable discussion and figure out solutions. The Product Owner can provide more details about the behavior of the selected item. The work planned in previous sprints, which has not been done, because of objectives reduction, becomes the priority for the next sprint.	Product Owner Scrum Team
	Estimating tasks	In order to distribute the development work between the team members, the duration of each task is estimated. Estimation is made in hours and taking into account that each task should be light	Scrum Team

(continued)

Table 3 (continued)

Activity	Task	Description	Involved roles
		enough(less than 16 hours). The team collectively addresses this task, reviewing the technical aspects during the discussion.	
	Assigning tasks	The team considers the number of persons required for each activity. All the activities must be studied, including *work meetings*, *coding*, and *document reviews*. It is desirable to delay the assignment of activities until the availability of the related team members is known.	Scrum Team
	Getting team commitment	This is a relevant task in which the team collectively decides what are the backlog items to implement in the current sprint. This decision constitutes the agreed and shared commitment of the sprint.	Scrum Team

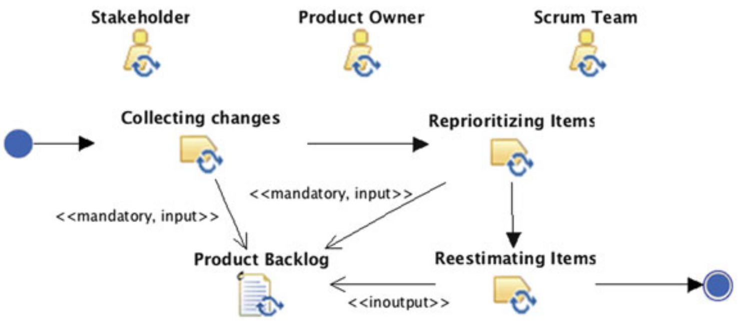


Fig. 18 Workflow of the *Update Product Backlog* activity

Table 4 Description of the tasks of the *Update Product Backlog* activity

Activity	Task	Description	Involved roles
Update *Product Backlog*	Collecting changes	Stakeholders or any member of the Scrum Team suggest new goals, functionalities, or changes to be added to the backlog. The bugs and enhancement requests obtained from product tests in the previous sprints are also incorporated into the collection of items that can be updated. The Product Owner decides whether to consider or not these items.	Product Owner Stakeholder Scrum Team
	Reprioritizing the items	New considered items must be prioritized.	Product Owner
	Reestimating items	New items added to the backlog have to be estimated by the Scrum Team, ideally during a brainstorming with the Product Owner previously to the next *Sprint Review*. The new estimation could imply a reestimation of existing items.	Scrum Team

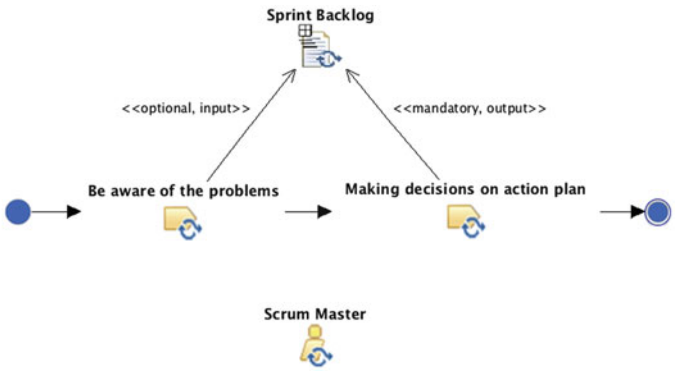


Fig. 19 Workflow of the *Manage Problems* activity

Table 5 Description of the tasks of the *Manage Problems* activity

Activity	Task	Description	Involved roles
Manage Problems	Be aware of the problem	The objective of the Scrum Master is to be informed about problems with the project. The daily meeting is the ideal moment to detect them. A good practice to follow up on the problem reporting is an informal chat initiated by the Scrum Master with the problem reporter.	Scrum Master
	Making decisions on the action plan	The Scrum Master must try to solve problems as soon as possible, in order to not disturb team progress. The master identifies possible solutions, and schedules meetings with owners if problems cannot be easily solved or need a management decision.	Scrum Master

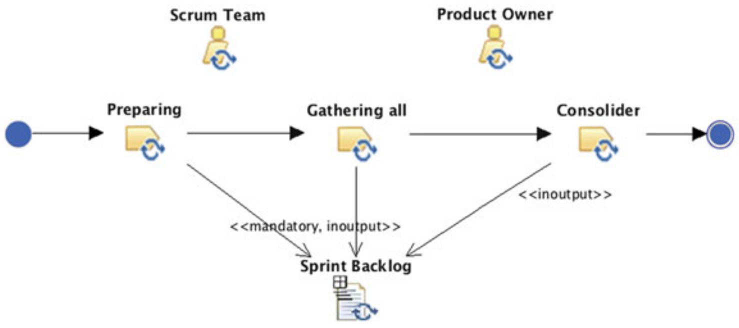


Fig. 20 Workflow for the *Conduct Scrum Daily Meeting* activity

Conduct Retrospective

The last activity of each sprint iteration is the *Sprint Retrospective*. This activity gives an opportunity to reflect at the end of the sprint and to identify the work to improve in the next sprint. The whole Scrum Team participates together with

Table 6 Description of the tasks of the *Conduct Scrum daily meeting* activity

Activity	Task	Description	Involved roles
Conduct Scrum daily meeting	Preparing	All the people update their planning (this sprint backlog) considering the remaining work for each task. As result of this task, an update or creation of a *Sprint Burndown Chart* is done.	Scrum Team
	Gathering all	Each team member reports her/his progress by answering the following 3 questions: *What has been done since the previous daily scrum*? *What will be done today*? *What problems have appeared*? Answers to the questions provide the update of the Sprint Backlog and changes in the Sprint Burndown Chart.	Scrum Team
	Consolidating	The resulting Sprint Backlog will help to take the right decisions on adjusting the Sprint Goal by removing forecasting content to be in time. Submitted problems are solved by the Scrum Master during the *Manage problems* activity.	Scrum Team

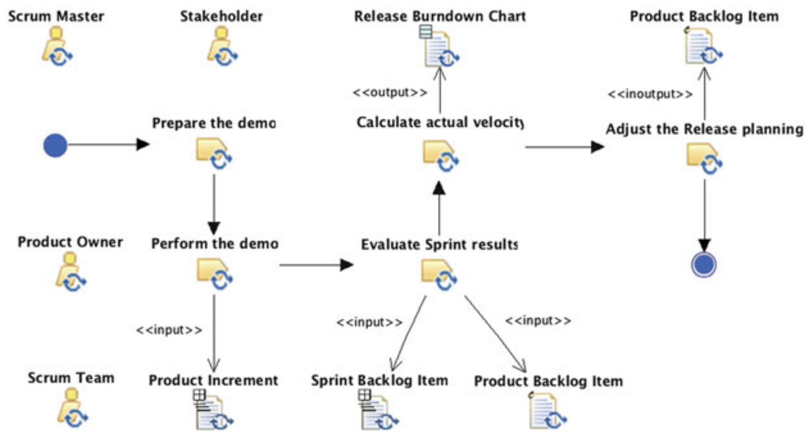


Fig. 21 Workflow for the *Review Sprint* activity

the Product Owner. The detailed tasks of the activity, its work products, and the collaborations between the roles are presented in Table 8 and Fig. 22. The complete activity has to be restricted to 2 h, though the average time is 1 h.

Release Work

As pointed previously, this activity is an optional one on the Sprint Phases. Its main objective is preparing a product release. Its actual execution depends on the way the product is made available to endusers and may vary from a project to another and

Table 7 Description of the tasks of the *Review Sprint* activity

Activity	Task	Description	Involved roles
Review Sprint	Preparing the demo	The schedule of the review is adjusted taking into account that the meeting must not exceed one hour.	Scrum Master
	Performing the demo	The team shows a demo of the product focused on the new goals satisfied during the actual sprint, allowing users to get a measurement of the work progress.	Scrum Team
	Evaluating the Sprint results	During this task, the Product Owner and the attending Stakeholders interview the team, provide impressions, make new proposals, and exchange requests. As a result, the Product Backlog is enhanced with the new items and with the bugs found.	Product Owner Stakeholders Scrum Master
	Calculating the actual *velocity*	A Release Burndown Chart is commonly used to compare the Sprint velocity with previous ones. The velocity is calculated as the sum of all items checked and approved as fully implemented.	Scrum Master
	Adjusting the *Release planning*	After reviewing the sprint, the team may realize that the initial conditions have changed from the last release planning. Then, it can be necessary to adjust it taking into account the items added, modified, or deleted, the changes in their priorities, the updated estimations, and of course, the average velocity of the team.	Scrum Team

Table 8 Description of the tasks of the *Conduct Retrospective* activity

Activity	Task	Description	Involved roles
Conduct retrospective	Discussion	Each team member is invited to express her/his bad and good results in the sprint providing solutions to the detected problems.	Scrum Team
	Defining the action plan	The team talks about potential improvements for the next sprint and the Scrum Master adds those approved to the product backlog. After that, the team sets up priorities with the help of the product owner and the stakeholder (if any).	Scrum Team Product Owner Stakeholder

among teams. Moreover, the team should rollout this optional activity with tasks that are not considered during "*normal*" sprints.

The only recommendation for this activity is trying not to make code changes because it is too late in the development process and it would imply a high risk of introducing bugs and errors in the final application. Some tasks that could be performed in this activity are:

- Hot deployment.
- Product packaging.
- Online download publishing.

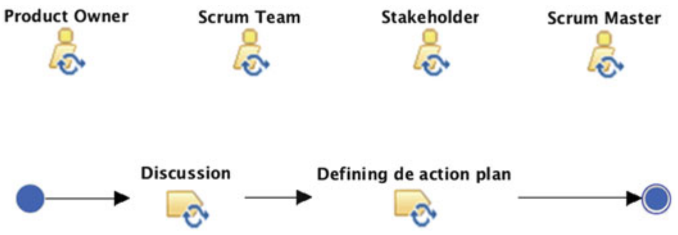


Fig. 22 Workflow for the *Conduct Retrospective* activity

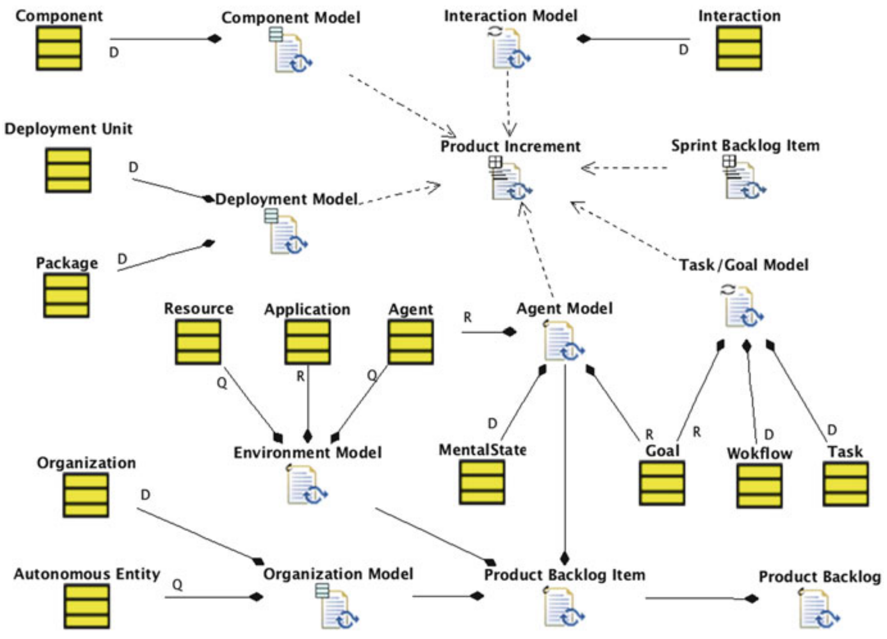


Fig. 23 The Sprint Work Product Model structure

- Technical documentation.
- User training.
- Product marketing.

2.2.3 Work Products

The Sprint Model generates several composed work products based on an INGENIAS specification. Their relationships with the MAS metamodel elements are described in Fig. 23.

Work Product Kinds

Table 9 describes the workproducts of each Sprint that are detailed in the next subsections.

Table 9 Work Product kinds for each Sprint phase

Name	Description	Work product kind
Product Increment	Source code for an increment of the product developed and finished on the previous or actual sprint	Structured
Product Backlog	A set of INGENIAS models that specify the product to obtain	Composited
Sprint Backlog	A set of PBI that the team must complete during the sprint	Composited
Burndown Chart	Graphical charts that show the evolution of the work	Structural
INGENIAS Model	See **IGENIAS-UDP chapter**	Structural,composite, behavioural

Product Increment

The primary work product of a Scrum project on each Sprint Phase is the *Product* itself. The Scrum Team is expected to bring the product or system to a potentially shippable state at the end of each Scrum Sprint. A product increment following the INGENIAS-Scrum approach is obtained from the IDK by generating code using the IAF. The code will be determined by the specified INGENIAS Models.

Product Backlog

During the iterations produced on the Sprints, each *Scrum Product Backlog* defined in the Preparation phase is completed and new ones are proposed and constructed. To complete the product backlog, the team must complete the life cycle of all its PBI reaching one after another the following states: *created*, *estimated*, *planned* (associated to a future sprint), *associated to current sprint* (implementation is ongoing), and *done*.

Sprint Backlog

As indicated in the previous sections, on the first day of a sprint and during the sprint planning meeting, team members create the *Sprint Backlog*. The sprint backlog can be thought of as the team's to-do list for the sprint. Whereas a product backlog is a list of features to be built, the sprint backlog is the list of tasks the team needs to perform in order to deliver the functionality they committed to deliver during the sprint.

Burndown Charts

Optional work products are the *Sprint Burndown Chart* and the *Release Burndown Chart*. Burndown charts show the amount of work remaining either in a Scrum sprint or a release. They are used for determining at a glance the evolution of a sprint or release, showing whether all the planned work will be finished by the desired date.

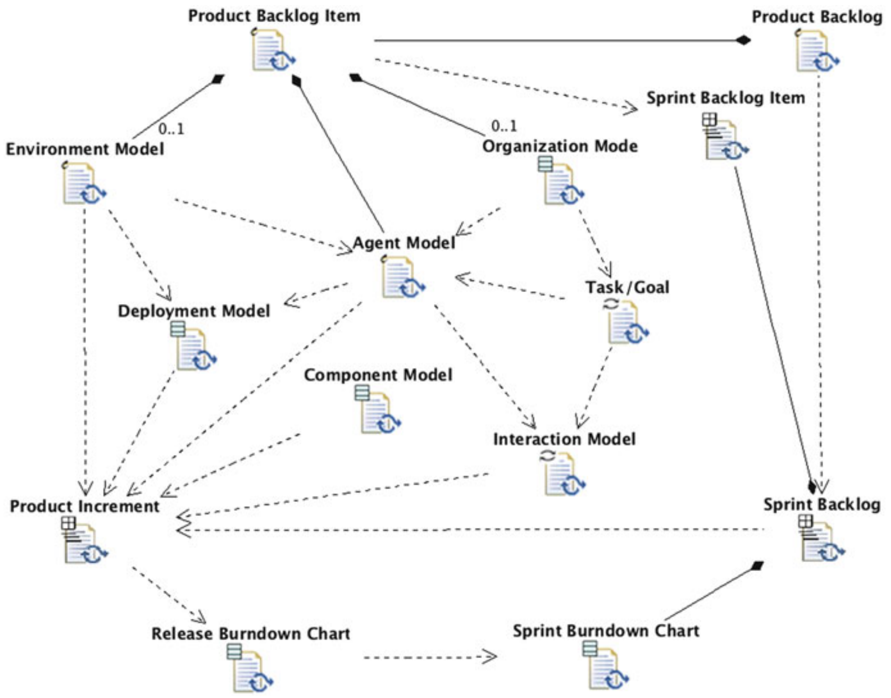


Fig. 24 Dependencies among INGENIAS-Scrum work products

INGENIAS Models

The INGENIAS-Scrum approach suggests additional work products for the process. The INGENIAS models are required in order to use the IDK and IAF tools for the automated code generation and test production. Nevertheless, part of the required code must be externally developed. For instance, some functionality related to resources and applications (e.g., graphical user interfaces or databases) and also concrete details of behavior (i.e., algorithms). INGENIAS does not provide modelling primitives to specify graphically these elements, though it has primitives that act as containers to embed this code in models. This allows that, afterwards, the IAF uses that information in the code generation. A developer may also find necessary or useful to modify the generated code.

2.3 Work Product Dependencies

Figure 24 shows an overview of the INGENIAS-Scrum work products, as well as their dependencies. As shown in the figure, the Agent Model depends on the Organization and Environment Models, and the Interaction Model has dependencies with the Agent and Tasks & Goals Models, among others. There are also dependencies between product backlogs (what to get), sprint backlogs (what to do), and Burndown charts (how much is done).

Regarding the structure of the work products associated to INGENIAS, they are fully detailed in the *INGENIAS-UDP chapter*.

3 Case Study: Conference Management System

The Conference Management System (CMS) is an interesting case study that involves several aspects, from the main organization issues to paper submission and peer review. These are typically performed by a number of people distributed all over the world who exploit the Internet as the infrastructure for communication and cooperation.

This section presents a solution to this problem following the INGENIAS methodology, and applying an adaptation of the Scrum management framework to define an agile development process. The proposed process makes use of the INGENIAS metamodel to define the system through several models, and of support model-driven tools to generate automatically code from the specifications. Following this approach, the development of the CMS implies an incremental and iterative life cycle based on performing two phases: the *Preparation* phase and the *Sprint* phase.

The *INGENIAS with the UDP chapter* in this book presents a similar solution to this problem. It follows the original proposal based on the UDP [14] and the Rational Unified Process (RUP) [17]. Some aspects of this chapter are referred to the previous one, where the reader can find a more detailed description.

3.1 Preparation Phase

As pointed out in Sect. 2.1, this phase comprises the activities: *Initiate Product Backlog*, *Preparation Tasks*, and *Plan Release*. The latter is done after completion of the remaining two, which are developed in parallel.

The initial Product Backlog is a preliminary description of the product requirements. In this approach, it is a specification conforming to the INGENIAS metamodel and generated with its tools. It can be developed from scratch or taking as basis examples provided with the IDK distribution.

In order to apply the first strategy, the team has to be familiarized with the IDK tool and know what examples the IAF provides with its distribution: *the hello world*, *a GUI agent example*, and *an interaction example*. Any of these examples can be used as starting point for this initial backlog, although they do not cover the complexity of the CMS.

The second approach is the one adopted here, as the CMS is a well-known and documented case study in the literature. This implies creating an initial *Organization Model* applying the three tasks offered by the original INGENIAS process: *Identify groups*, *Generate group members*, and *Identify goals*. Figure 25 shows the resulting

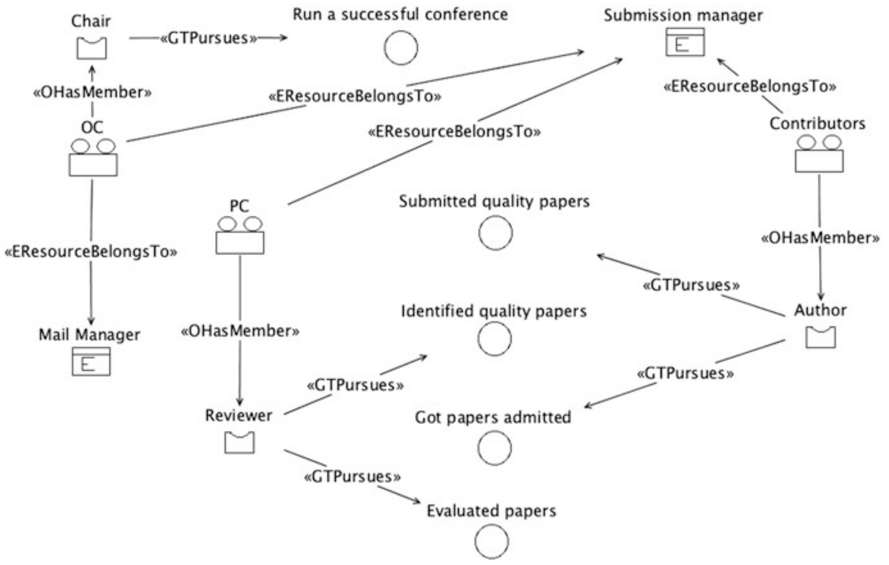


Fig. 25 Organization diagram with the main user groups of the CMS and their access to applications

model. Note that the result is just the same than the one obtained using the original INGENIAS development process, but without generating use cases in order to identify the functionality perceived by the user.

This first activity identifies three groups in the CMS case study. The *OC* (Organizing Committee) includes the *Chairs*, the *PC* (Program Committee) for the *Reviewers*, and the *Contributors* with the *Authors*. These groups perform tasks related to certain goals. The main goal that the *OC* must satisfy is *Run a successful conference*. Regarding the *PC*, it has to satisfy the goals: *Identified quality papers* and *Evaluated papers*. Finally, *Contributors* have to satisfy the goals *Submitted quality papers* and *Got papers admitted*. There are also two external applications, that is, the *Submission manager* and the *Mail manager*. Group members use them to get some services, according to certain use constraints. Every group can access the *Submission manager*, though not for the same tasks, but only the *OC* can use the *Mail manager*.

The Product Owner prioritizes the goals to be addressed in the sprints in order to fix the *Plan Release*. In this case, the choice is to consider in the first sprints the behavior associated to the *Run a successful conference* goal, which is associated to the *OC* group. The assistants to the meeting agree that the goals associated with the roles *Reviewer* and *Author* can be done in later sprints because the associated risk is lower than the selected one.

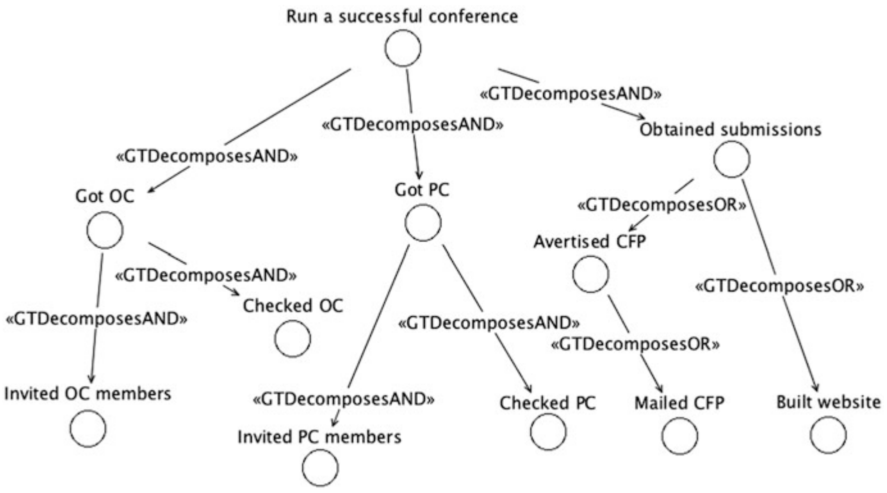


Fig. 26 Initial decomposition of the *Run a successful conference* goal

3.2 Sprint Phase

In the first sprint, the Scrum team performs the *Daily Works* activity related with the *Run a successful conference* goal. Its first results are the diagrams in Figs. 26 and 27 for the Tasks and Goals Model, and Fig. 28 for the Agent Model. Figure 26 shows the refinement of the selected goal in different subgoals. Figure 27 introduces the tasks needed to achieve those goals, that is, their products are potentially able to satisfy the conditions related to those goals. Finally, Fig. 28 shows the initial Agent Model obtained from the first refinement of the Organization Model, assigning the identified tasks to the *Chair* role.

The *Daily Works* activity follows with the INGENIAS activity cycle that includes the tasks: *Show task execution*, *Generate a Component Model*, *Generate a Deployment Model*, *Specify code templates to apply*, and *Validate code*. Regarding these tasks, the team follows short iteration cycles that give a similar result to those explained in Sect. 3 in the *INGENIAS chapter for the Construction Phase*. Anyway, the last activities are beyond the scope of this case study, which is focused on the development process adopted and not on the application obtained. Nevertheless, the facilities included in the IDK [11], and in particular the IAF [12], facilitate generation of testing code for these specifications. These tests will implement the acceptance tests suggested for each BPI associated with the specified goals.

The product obtained is used in the *Scrum Daily Meeting* and *Review Sprint* activities. At the end of each sprint, the *Conduct Retrospective* activity is used to better accomplish the remaining PBI in the future sprints through their revision regarding the results of the current sprint.

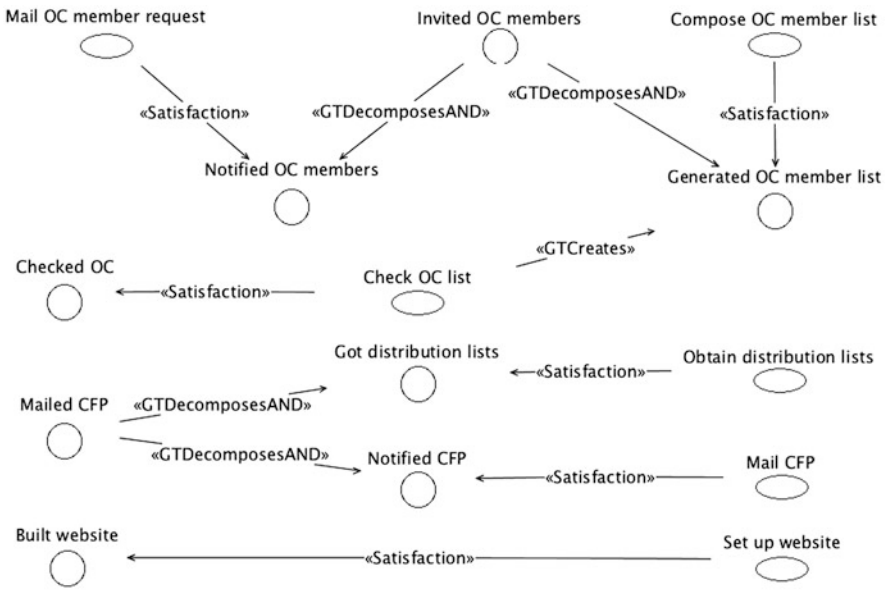


Fig. 27 Initial assignment of tasks related to the *Run a successful conference* goal

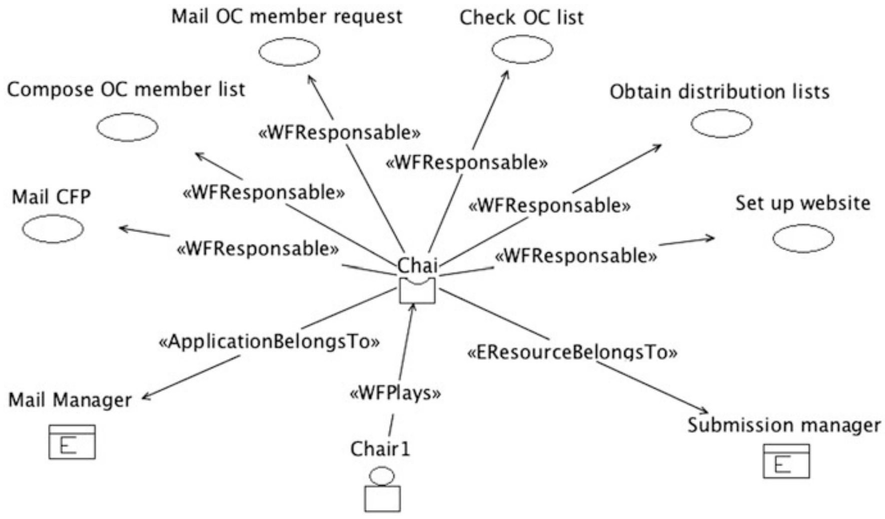


Fig. 28 Initial assignment of tasks to the *Chair* role

References

1. García-Magariño, I., Gómez-Rodríguez, A., Gómez-Sanz, J., González-Moreno, J.C.: Adv. Soft Comput. **50**, 108 (2009)
2. Cernuzzi, L., Cossentino, M., Zambonelli, F.: Eng. Appl. Artif. Intell. **18**(2), 205 (2005)
3. Martin, R.: Agile Software Development: Principles, Patterns, and Practices. Prentice Hall PTR, Upper Saddle River (2003)
4. Schwaber, K., Beedle, M.: Agile Software Development with Scrum. Prentice Hall, Upper Saddle River (2001)
5. Schwaber, K.: In: 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA 1995), pp. 117–134 (1995)
6. Pavón, J., Gómez-Sanz, J.J., Fuentes-Fernández, R.: In: Henderson-Sellers, B., Giorgini, P. (eds.) Agent-Oriented Methodologies. Article IX, pp. 236–276. Idea Group Publishing, Hershey (2005)
7. France, R., Rumpe, B.: In: 2007 Future of Software Engineering (FOSE 2007), pp. 37–54. IEEE Computer Society, Minneapolis (2007)
8. García-Magariño, I., Fuentes-Fernández, R., Gómez-Sanz, J.: Inf. Softw. Technol. **51**(8), 1217 (2009)
9. Fuentes-Fernández, R., Gómez-Sanz, J.J., Pavón, J.: IEICE Trans. Inf. Syst. **E90-D**(8), 1243 (2007)
10. Fuentes-Fernández, R., Gómez-Sanz, J.J., Pavón, J.: Int. J. Agent Oriented Softw. Eng. **1**(1), 2 (2007)
11. Gómez-Sanz, J.J., Pavón, J., Fuentes-Fernández, R., García-Magariño, I., Rodríguez-Fernández, C.: INGENIAS Development Kit, V. 2.8. Tech. rep., Universidad Complutense de Madrid (2008)
12. Gómez-Sanz, J.: INGENIAS Agent Framework. Development Guide V. 1.0. Tech. rep., Universidad Complutense de Madrid (2008)
13. Bellifemine, F., Poggi, A., Rimassa, G.: In: 5th International Conference on Autonomous Agents (AGENTS 2001), pp. 216–217. ACM, Montreal (2001)
14. Booch, G., Jacobson, I., Rumbaugh, J.: The Unified Software Development Process. Addison-Wesley, Reading (1999)
15. Gómez-Sanz, J.: Modelado de sistemas multi-agente. Ph.D. thesis, Universidad Complutense de Madrid, Facultad de Informática (2002)
16. Grupo de Investigación en Agentes Software: Ingeniería y Aplicaciones. INGENIAS Section. http://grasia.fdi.ucm.es/main/?q=es/node/61 (2010)
17. Rational Software. Rational Unified Process: White Paper (1998)