An Agent-based Approach for the Management of Distributed Workflows

Giancarlo Fortino

g.fortino@unical.it

LISDIP (Laboratorio di Ingegneria dei Sistemi Distribuiti e Paralleli)
DEIS - University of Calabria, ITALY

Agentlink III - AOSE TFG - Budapest 15-17 Sept. 2005

Motivations and Goal

- Workflow management systems are being subjected to on-going fervent development not only in their traditional application area, namely business process management, but also in data-intensive scientific and engineering application areas which demand for an effective modeling and an efficient management of distributed workflows.

- To effectively fulfill such requirements the emerging *Agents* paradigm and technology can be employed.

- Agents have already been used in WFMS mainly as: (i) personal assistants; (ii) case managers; (iii) middleware components

Proposal

- Differently from most of past/current efforts, more focused on technology, we propose an agent-based approach covering modeling and enactment so exploiting the enormous potential of *Agents* in completely supporting the development of complex systems from their modeling to their implementation.
  - The modeling phase is enabled by a MAS (Multi-Agent System) meta-model for the management of workflows which can be instantiated by a workflow schema based on the Workflow Patterns
  - The enactment phase is supported by an agent-based enactment framework designed on the basis of the MAS meta-model so allowing a seamless transition from modeling to enactment of distributed workflows

Outline

- Basic concepts on workflow management
- A MAS-meta model for the management of distributed workflows
- Examples of instantiation of the MAS-meta model
- An agent-based framework for the enactment of distributed workflows

Workflow Management Systems

- Workflow management systems (WFMS) are systems designed to automate complex activities consisting of many dependent tasks.

- A reference model for WFMS, proposed by the Workflow Management Coalition (WfMC), defines the basic architecture of a WFMS as consisting of several components:
  - workflow/process definition module, administration and monitoring tools, workflow client application, invoked applications and external workflow engines

  which are interfaced with the workflow enactment service.

- The workflow definition module provides the workflow designer with a workflow language able to specify a workflow schema which can be successively instantiated by means of the enactment service based on a workflow API and on one or more workflow engines

Workflow Management Systems

- The workflow definition language is to be expressive and powerful to specify complex workflows from several perspectives: *control-flow*, *data-flow*, *resource* and *operational*.

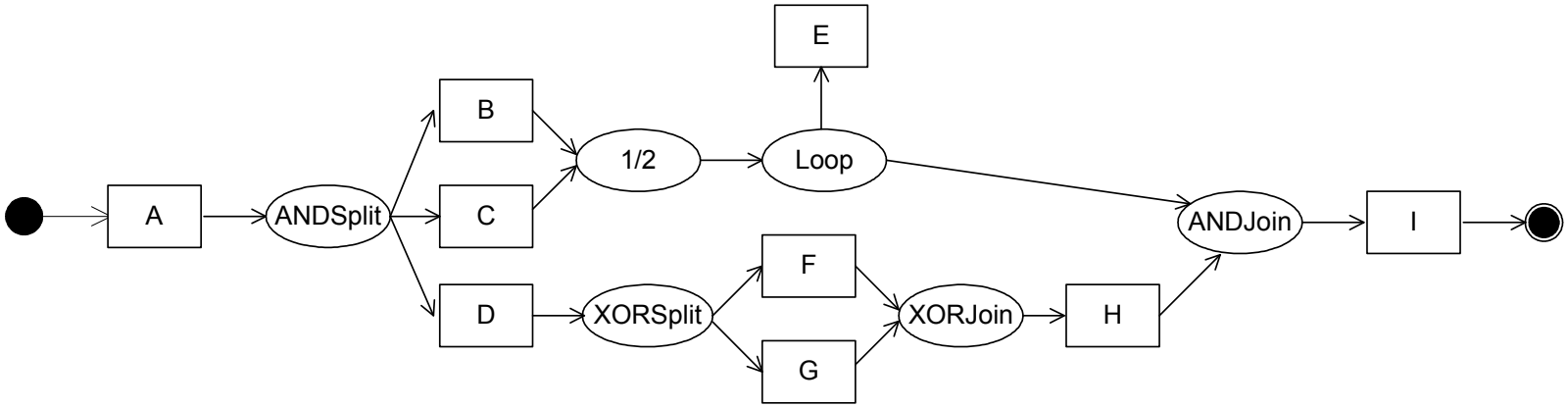
- A workflow schema from the control-flow perspective can be formalized by the tuple $\mathcal{WFS}=<T, E, t_0, T_F>$, where:
  - $T=T_X \cup T_R$ is a finite set of tasks partitioned into execution tasks T_X and routing tasks T_R,
  - $E \subseteq (T-T_F)\times(T-\{t_0\})$ is a relation of precedence among tasks,
  - $t_0 \in T_R$ is the initial pseudo task,
  - $T_F \subseteq T_R$ is the set of final pseudo tasks.

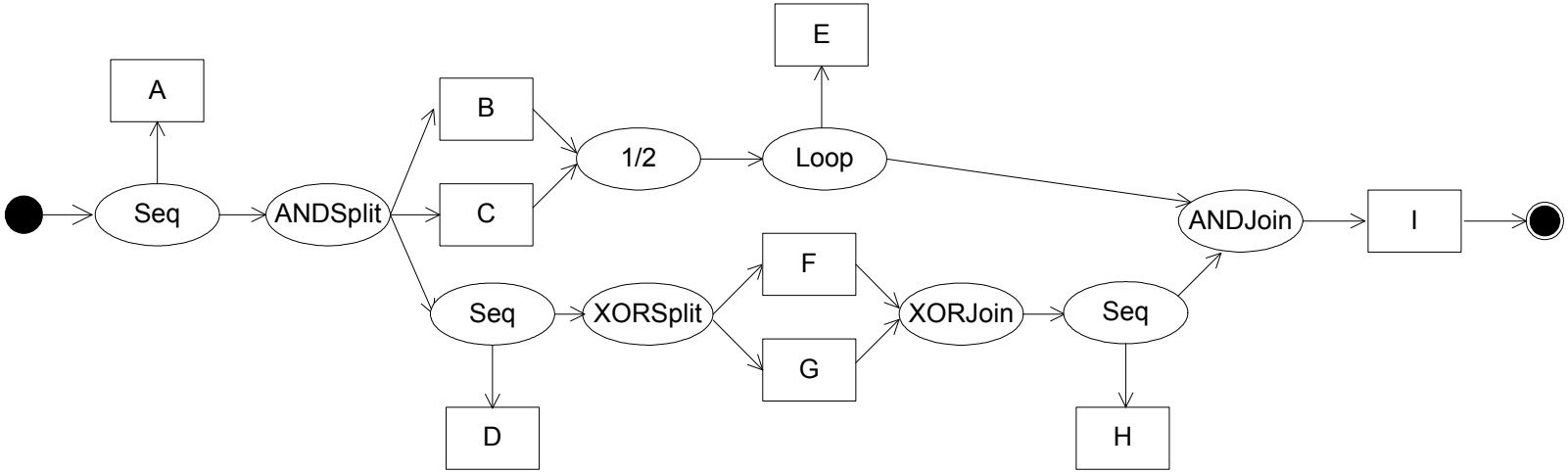
- For our purposes, the set of the routing tasks is the set of the available Workflow Patterns (WP) identified by Van der Aalst et al. which represent an expressive and powerful set of control-flow constructs.

Workflow Management Systems

PATTERN TYPE	PATTERN NAME (*SYNONYMS*)	DEFINITION
Basic Control Flow	Sequence (*Sequential routing, serial routing*)	A task is enabled after the completion of another task.
	Parallel Split (*AND-split, parallel routing, fork*)	A point in the workflow where a single thread of control splits into multiple threads of control which can be executed in parallel, thus allowing tasks to be executed simultaneously or in any order.
	Synchronization (*AND-join, rendezvous, synchronizer*)	A point in the workflow where multiple parallel tasks converge into one single thread of control, thus synchronizing multiple threads.
	Exclusive Choice (*XOR-split, conditional routing, switch, decision*)	A point in the workflow where, based on a condition, one of several branches is chosen.
	Simple Merge (*XOR-join, asynchronous join, merge*)	A point in the workflow where two or more alternative branches merge without synchronization.
Advanced Branching and Synchronization	Multi-choice (*Conditional routing, selection, OR-split*)	A point in the workflow where, based on a condition, a number of branches are chosen.
	Synchronizing Merge (Synchronizing join)	A point in the workflow where multiple paths converge into one single thread.
	Multi-merge	A point in a workflow where two or more branches reconverge without synchronization. If more than one branch gets activated the task following the merge is started for every activation of every incoming branch.
	Discriminator (*N/M or partial join*)	A point in a workflow that waits for a number of the incoming branches to complete before activating the subsequent task; then it waits for the remaining branches to complete and "ignores" them. Then it resets itself.
Structural	Arbitrary Cycles (*Loop, iteration, cycle*)	A point in a workflow where one or more tasks can be done repeatedly.
	Implicit Termination	A given sub-workflow should be terminated when there is nothing else to be done.
Multiple Instances	Multiple Instances without synchronization (*Multi-threading without synchronization, spawn off facility*)	Multiple instances of a task can be created with no need to synchronize them.
	Multiple Instances with a priori design time knowledge	A task is enabled a number of times known at design time. Once all instances are completed some other task needs to be started.
	Multiple Instances with a priori run-time knowledge	A task is enabled a number of times known at run time. Once all instances are completed some other task needs to be started.
	Multiple Instances without a priori run-time knowledge	A task is enabled a number of times known neither at design time nor at run-time. Once all instances are completed some other task needs to be started.
State-based	Deferred Choice (*External choice, implicit choice, deferred XOR-split*)	It is similar to the exclusive choice but the choice is not made explicitly and the run-time environment decides what branch to take.
	Interleaved Parallel Routing (*Unordered sequence*)	A set of tasks is executed in an arbitrary order decided at run-time; no two task are active at the same time.
	Milestone (*Test arc, deadline, state condition, withdraw message*)	The enabling of a task depends on the workflow being in a given state, i.e. the task is only enabled if a certain milestone has been reached which did not expire yet.
Cancellation	Cancel Activity *(Withdraw activity)*	An enabled task is disabled, i.e. a thread waiting for the execution of a task is removed.
	Cancel Case (*Withdraw case*)	A workflow instance is removed completely.

Workflow Management Systems





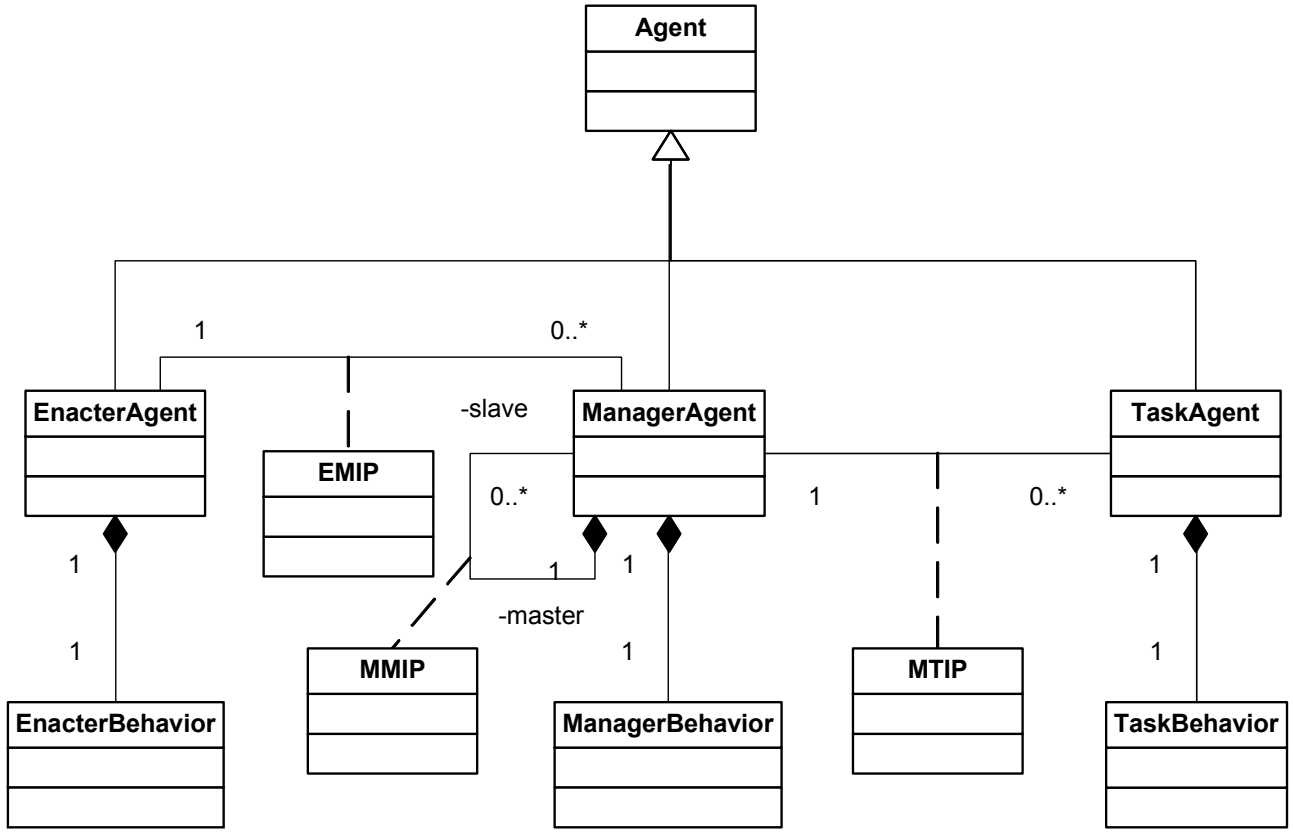
The MAS Meta-Model for Workflow Management

- A MAS meta-model for the management of distributed workflows should formalize the types, the acquaintance relationships and the behaviors of the agents involved in the activation, execution and monitoring of a distributed workflow schema.

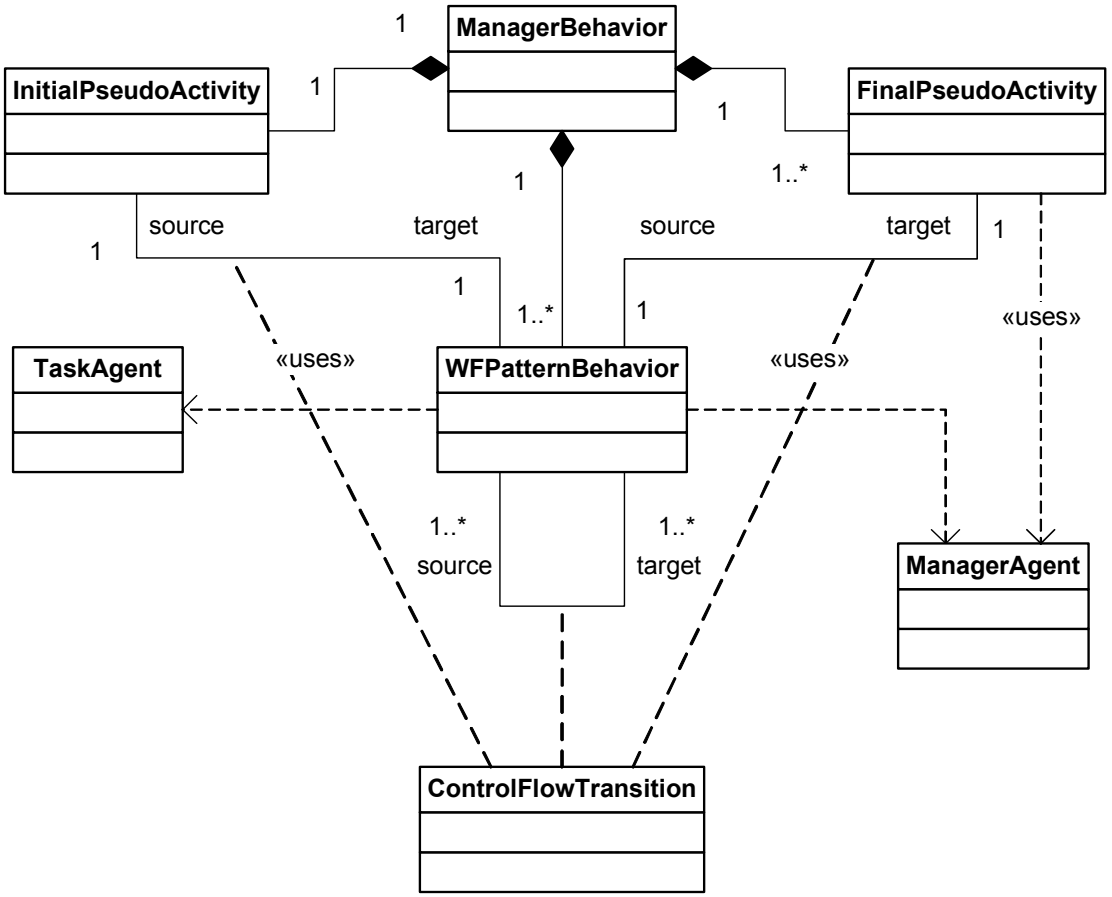
- The instantiation of the MAS meta-model should produce a MAS model able to enact the workflow schema used to instantiate the MAS meta-model.

- A produced MAS model is therefore the agent-based enactment engine model for the used workflow schema.

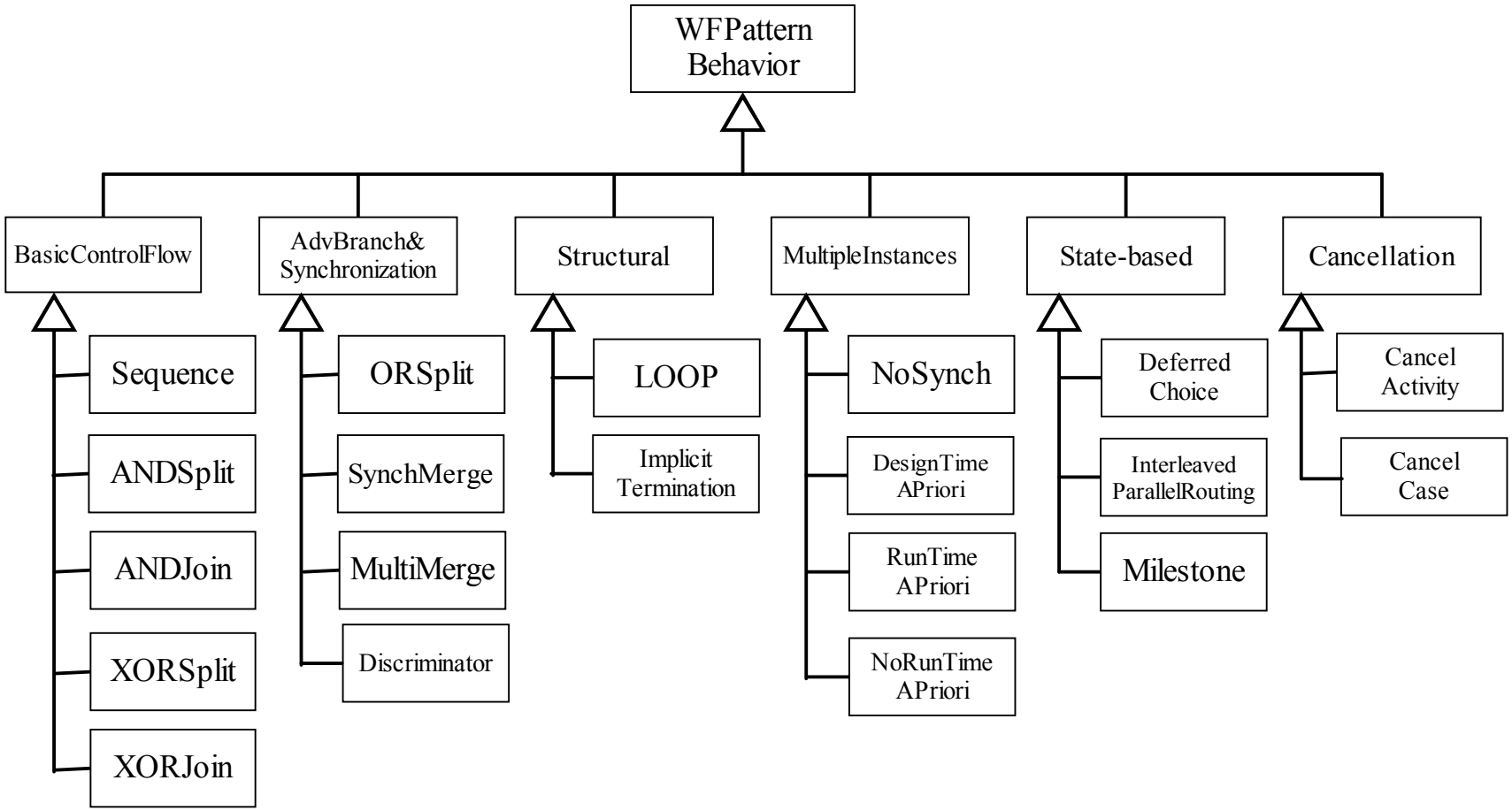
The MAS Meta-Model for Workflow Management: *Structural*



The MAS Meta-Model for Workflow Management: *Behavioural*



The MAS Meta-Model for Workflow Management: *Behavioural*



Flat Workflow Management

- The flat workflow management is enabled by a single *ManagerAgent* which embodies the schema of the workflow to enact.

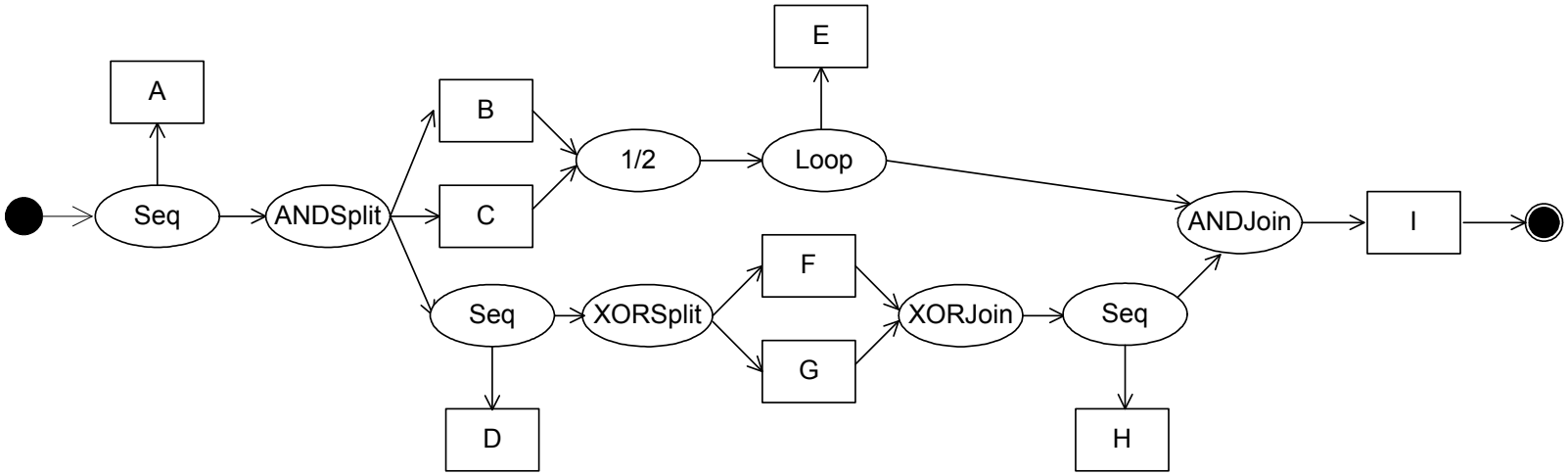
- Given a workflow schema $\mathcal{WfS}$, the MAS model able to enact $\mathcal{WfS}$ is obtained through the folllowing two-step procedure:

1. Construction of the *Structural* part of the MAS model. The *EnacterAgent* is linked to the *ManagerAgent* which is, in turn, linked to all the *TaskAgent*s corresponding to the execution tasks of the $\mathcal{WfS}$.

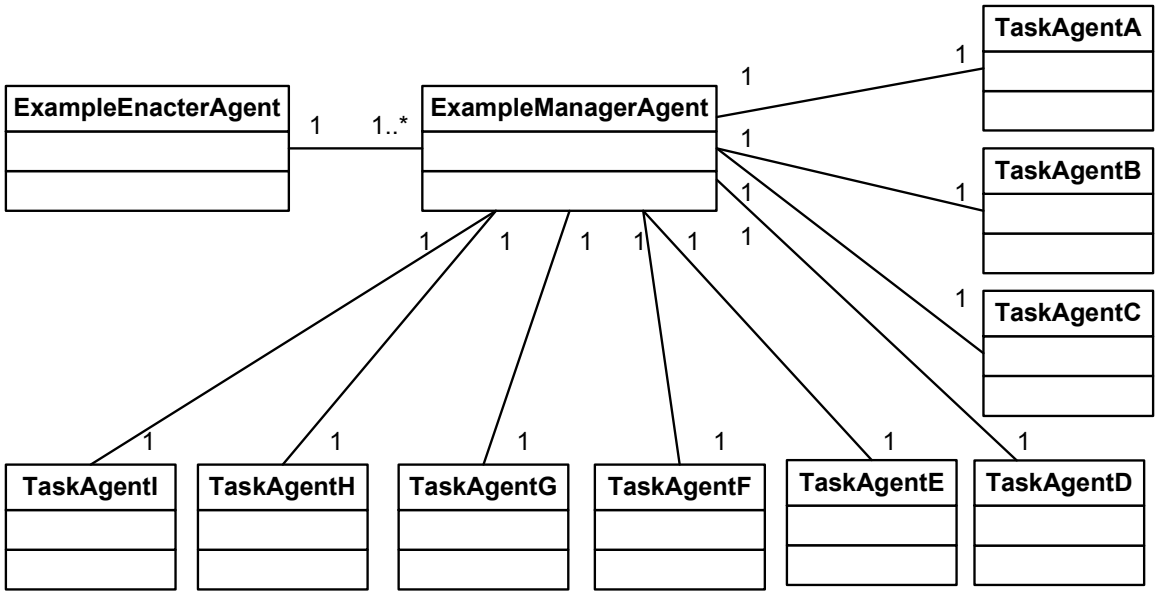
2. Construction of the *Behavioral* part of the MAS model. The *ManagerBehavior* is obtained by translating $\mathcal{WfS}$ in an interconnection of one *InitialPseudoActivity*, one or more *FinalPseudoActivity*, and one or more *WFPatternBehavior*.

   In particular, for each routing task (or workflow patterns) of $\mathcal{WfS}$, one class of the same type of the routing task is to be defined which has a *uses* association with the *TaskAgent*s related to the execution tasks to which the routing task is associated in $\mathcal{E}$. The defined classes are interconnected according to the precedence relation $\mathcal{E}_{\mathcal{R}} \subset (T_R - T_F) \times (T_R - \{t_0\})$ between the routing tasks.

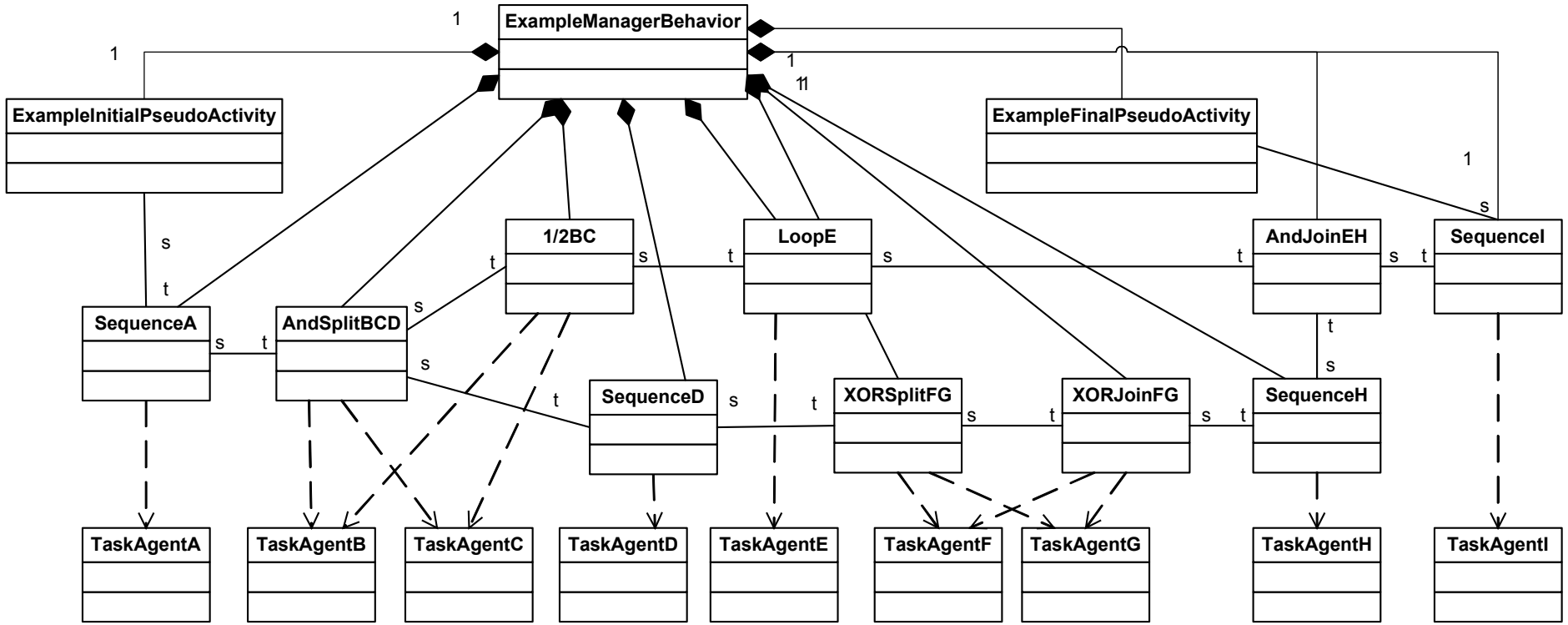
Flat Workflow Management



Flat Workflow Management: *Structural* layer of the MAS model



Flat Workflow Management: *Behavioural* layer of the MAS model

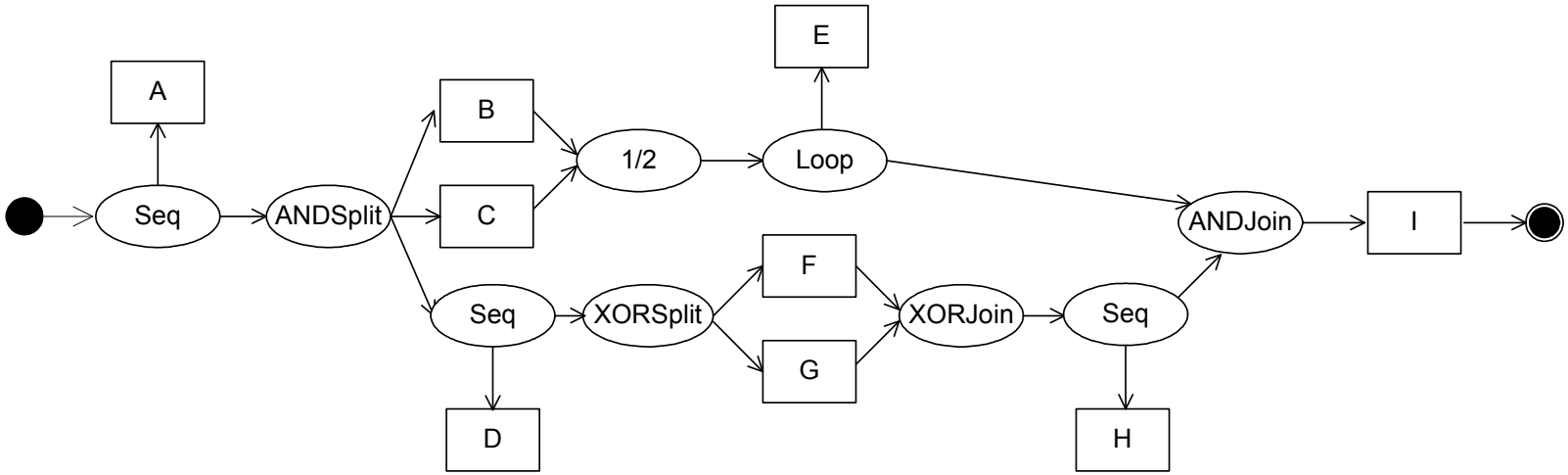


Hierarchical Workflow Management

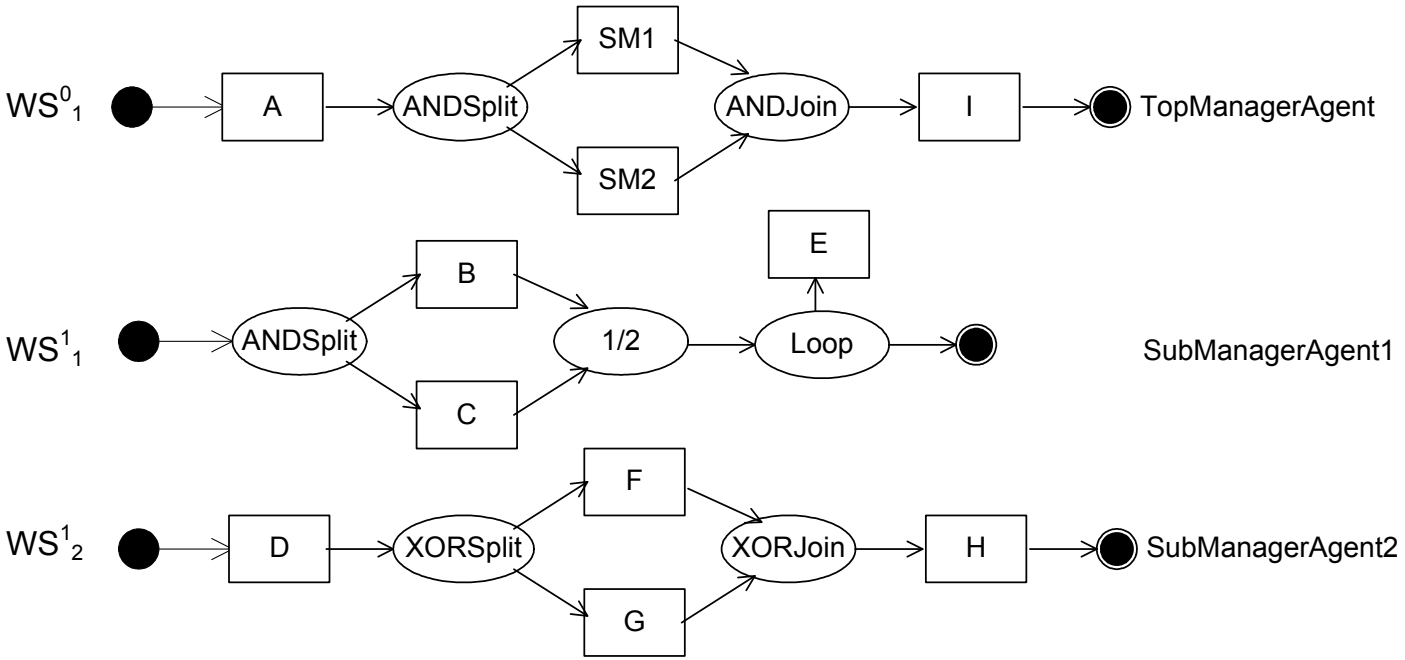
- The hierarchical workflow management is enabled by a set of *ManagerAgent*s each of which embodies a sub-schema of the workflow schema according to a hierarchical model.

- Given a workflow schema $\mathcal{WFS}$, the MAS model able to enact $\mathcal{WFS}$ is obtained through a three-step procedure:
  1. Partitioning of the $\mathcal{WFS}$ into a set of hierarchically arranged workflow schemas
  2. Construction of the *Structural* part of the MAS model
  3. Construction of the *Behavioral* part of the MAS model by specifying all the *ManagerBehavior*s

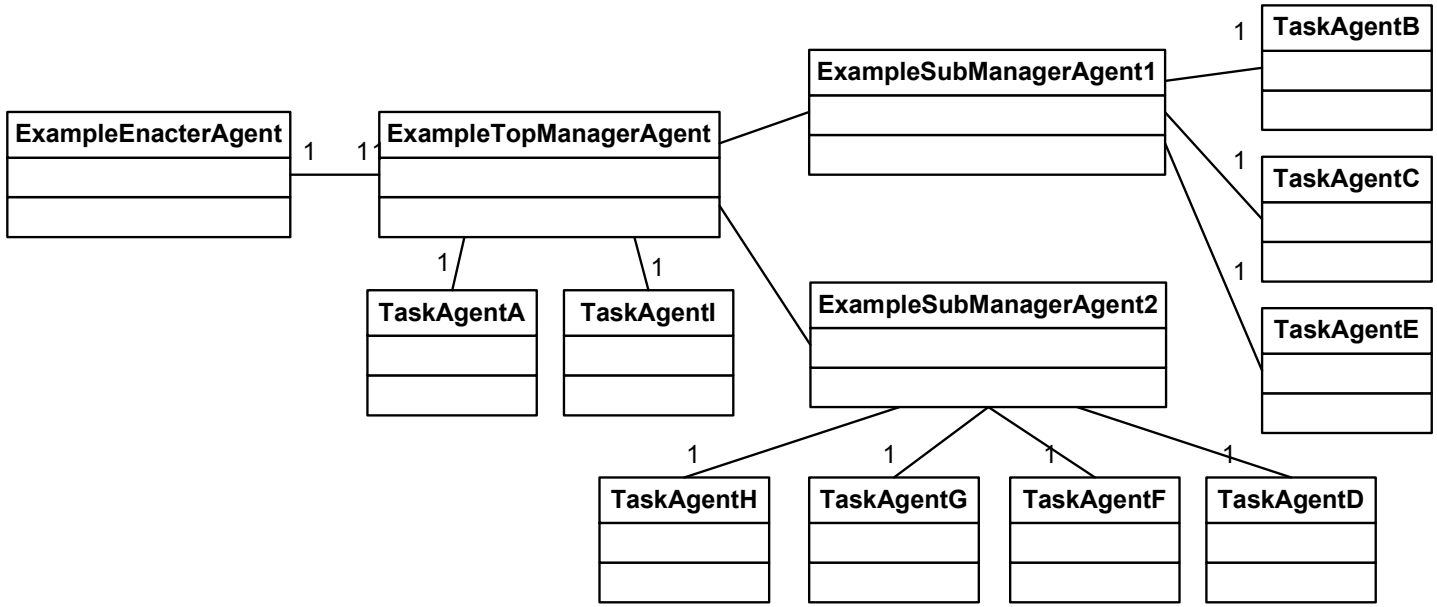
Hierarchical Workflow Management



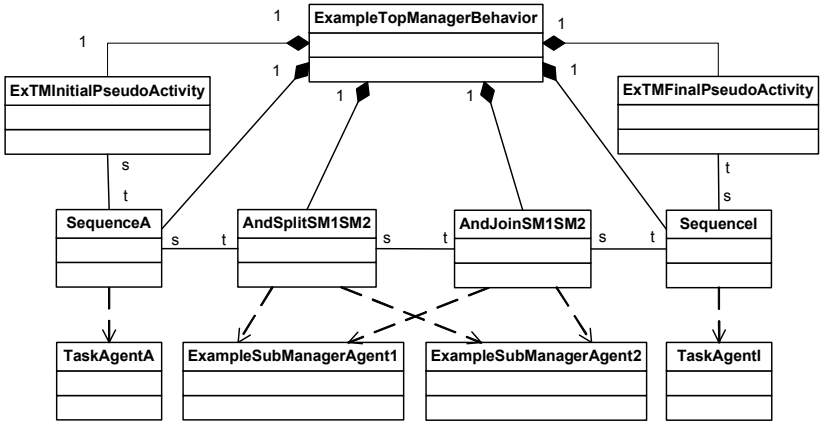
Hierarchical Workflow Management

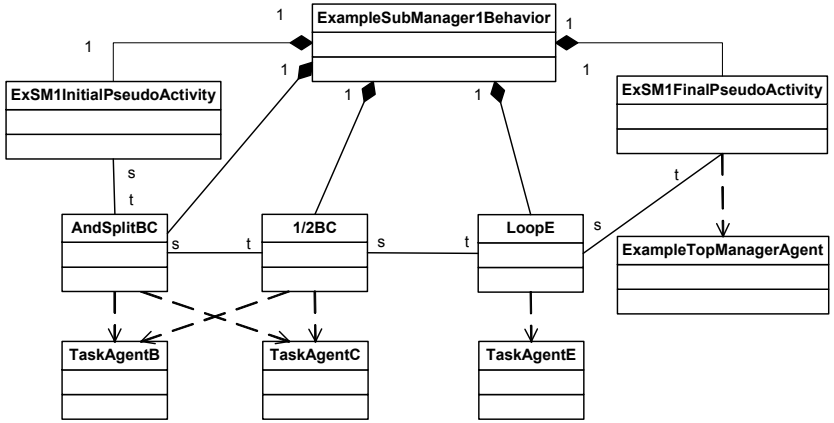


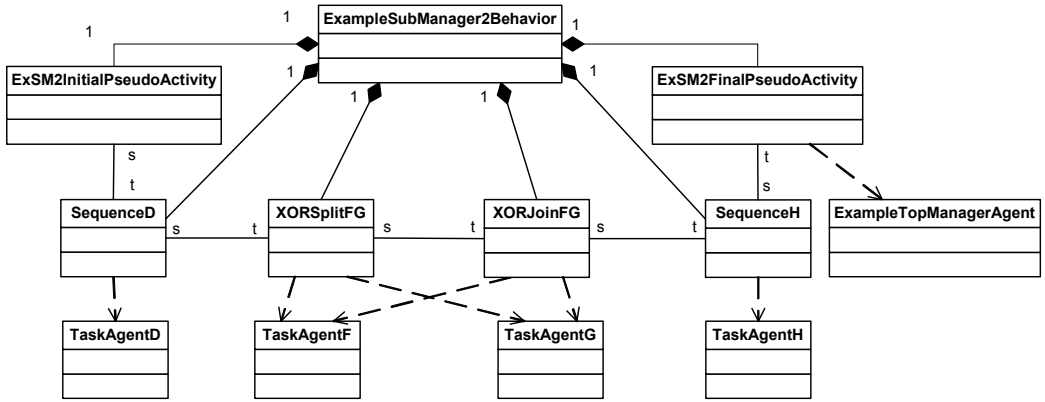
Hierarchical Workflow Management: *Structural* layer of the MAS model



Hierarchical Workflow Management: *Behavioural* layer of the MAS model







An Agent-based Framework for the Enactment of Distributed Workflows

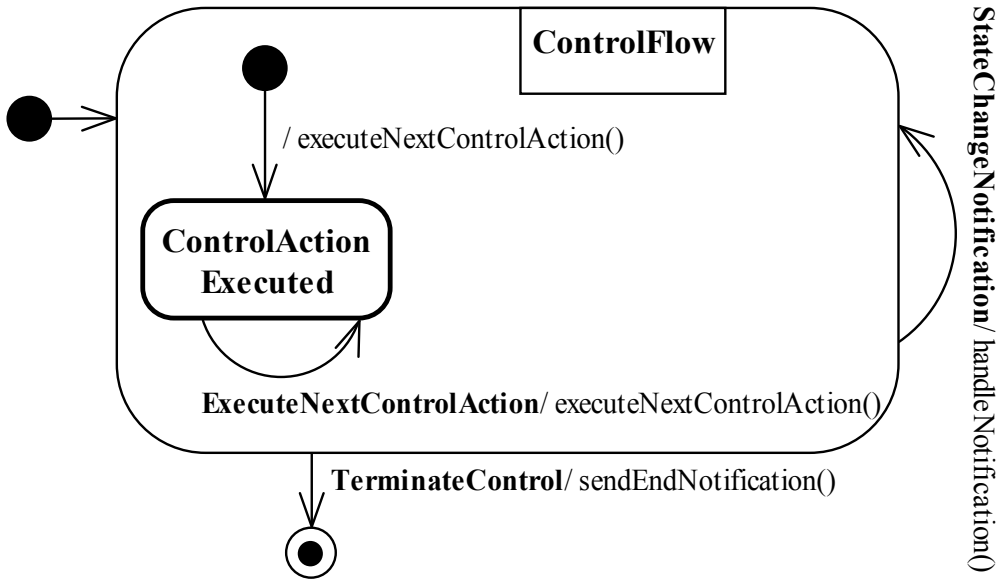
- The aim of an agent-based workflow enactment framework is to support the implementation of general workflow enactment engines or specific workflow systems.
- The structure of the framework, i.e. the base agent and agent behavior classes, is directly designed upon the MAS meta-model classes so exploiting the defined instantiation rules for flat and hierarchical workflow management.
- The dynamics of the framework are defined by appositely specifying the interaction protocols among the agent types (agent interaction protocols) and the dynamic behaviors of the workflow patterns (control-flow behavior).

An Agent-based Framework for the Enactment of Distributed Workflows

- Agent interaction protocols based on FIPA
  - *The Enacter/Manager Interaction Protocol (EMIP)*
  - *The Manager(master)/Manager(slave) Interaction Protocol (MMIP)*
  - *The Manager/Task Interaction Protocol (MTIP).*
- Control-flow behavior based on Distilled Statecharts
  - General control-flow behavior of the *ManagerAgent (ManagerBehavior)*
  - Specific control-flow behavior of each workflow pattern classes belonging to the *WFPatternBehavior* hierarchy

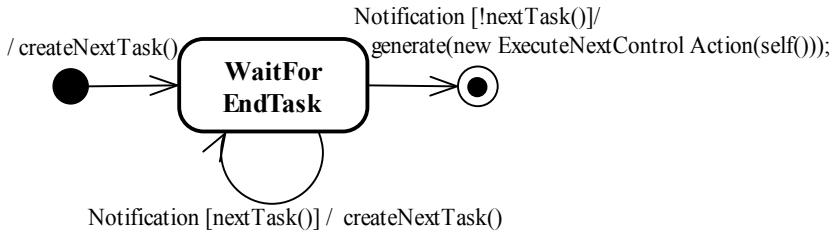
An Agent-based Framework for the Enactment of Distributed Workflows

ManagerBehavior

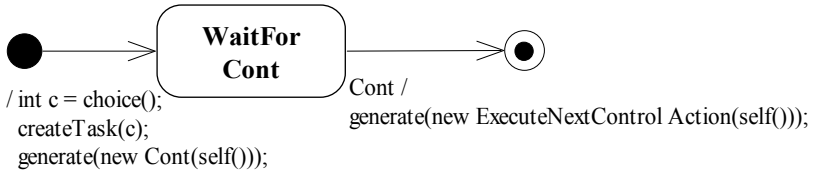


An Agent-based Framework for the Enactment of Distributed Workflows

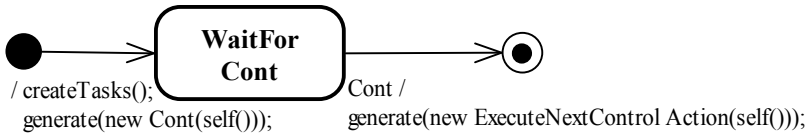
Sequence



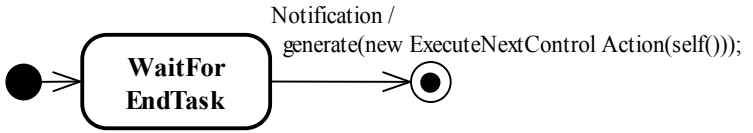
XORSplit



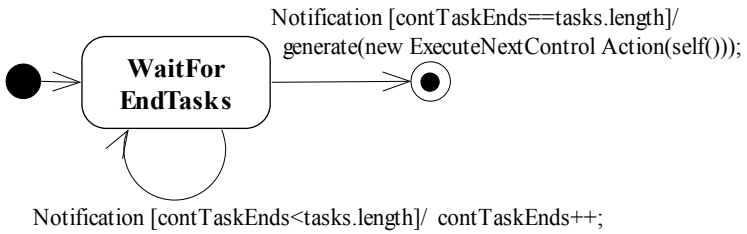
ANDSplit



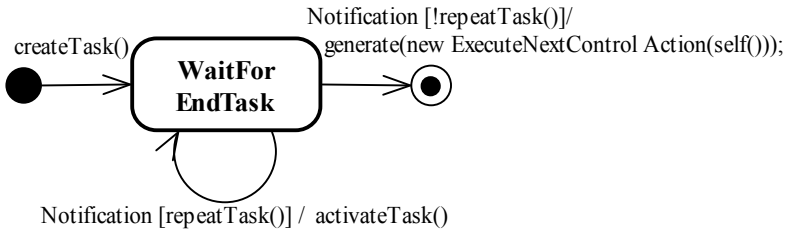
XORJoin



ANDJoin



Loop



An Agent-based Framework for the Enactment of Distributed Workflows

- Java-based Implementation
  - JADE-based
    - The class diagram of the JADE-based framework is an extension of the MAS meta-model class diagram
    - The agent classes (EnacterAgent, ManagerAgent, TaskAgent) extend the Agent class of JADE
    - The agent behavior classes (EnacterBehavior, ManagerBehavior, TaskBehavior) and the classes of the WFPatternBehavior hierarchy extend the Behaviour class or the classes (OneShotBehaviour, FSMBehaviour) deriving from the Behaviour class of JADE
    - The agent interaction protocols (EMIP, MMIP, MTIP) are appositely defined through sequences of ACL messages, instances of the ACLMessage class
  - ActiWare-based

Conclusions and Future Work

- The approach has been successfully applied to the implementation of a distributed workflow system for the monitoring of distributed agro-industrial productive processes.
- The efficacy of the modeling phase results from the synergic exploitation of the Agents paradigm and the Workflow Patterns, whereas the flexibility and robustness of the enactment phase mainly depend on the JADE-based implementation of the agent-based enactment framework.
- The results obtained here establish the basis for the construction of an agent-based general-purpose WFMS. Efforts are currently underway to develop such a system which is able to validate workflow schemas based on the Workflow Patterns and to enact them on a distributed platform based on JADE or ActiWare.