Università degli Studi di Bologna

Facoltà di Ingegneria e Scienze Informatiche
Corso di Laurea in Ingegneria

Relazione

Workflow, coordination and autonomy - Autonomous systems Exam

Candidato:
Emilio Fimiani
Matricola 744599

Relatore:
Ch.mo Prof. Andrea Omicini

Anno Accademico 2015-2016

Contents

Chapter 1

Introduction

As well as biological systems also computational systems process evolution generates improvement and progress towards increasing degrees of autonomy. Usually, arising complexity is related to this aspect and has to be managed. Complex systems are distributed, pervasive, intelligent, non-deterministic and self-organised. These characteristics have to be carefully and deeply handled to improve technology. Speaking of which, it was developed a very interesting approach that aims to understand, capture and bring natural system features to computational systems: nature-inspired Computing (NIC). Basically, it allows to define nature-inspired model for coordination and Self-organised Systems (SOS). Obviously, it's mandatory to know and understand natural systems structures, interactions, behaviours and processes in such a way to effectively apply biological system fundamentals to computational systems . The way biological systems integrate all regulatory and molecular processes is called Systems Biology. It has an analytical approach and allows to understand individual biological processes. Process, as stated in my paper [7], is a set of rationally linked tasks through which an organization generates value to reach its targets. A way of looking processes is the workflow (wf): units of work within a process generating products/services related to customer satisfaction. On one hand, workflow management helps organization to specify, analyze, execute and monitor the workflow. On the other hand wf management technology is a set of steps (actions), sequencing (order of actions), routing (who does the action) and conditions (rules for doing actions, routing actions) defined by a flowchart generator. These elements are no more than parts of a natural system and they have to be integrated to set up a system.

Notes

Part of the information in this document are taken from [16] and [7].

Chapter 2

Coordination for complex systems

Coordination is the science of managing the space of interaction: individuals coordinate actions each other, concurring to its completion. Actually, global properties of complex coordinated systems like natural systems depend on interactions that can be force through coordination model.

2.1 Living System, a complex system

Biology is the study of living systems or organisms: structure, growth, origin, evolution and distribution. Living systems are autopoietic units that sustain themselves on a inner network of reactions and continue to mantain the organised bounded structure (self-determined flexibility) that is an element of system identity (i.e. membrane cell). Autopoiesis is the ability of a complex system of mantaining its own overall coherence, structure and organisation, through the mutual interactions of its components. It is an explication of the autonomy of the living autonomous systems, specifying their own rules of behaviour, activities and domains of interaction. Every living system is composed by autonomous entities: cells. This system internal basic unit is called agent in computational systems.

Autonomy The term autonomy derives from greek: Auto + nomos, having its own laws. In fact, it defines homeostatic functions to regulate its internal conditions and keep functions stable (i.e. robustness, allows to mantain its funciton against internal/external perturbations). Autonomous systems are endegenous, self-organised, non-deterministic (unpredictable), goal-orientented and indipendent.

2.2 Agent as system internal basic unit

Agent (the one who acts) is proactive (making something happens) and autonomous. It hasn't an interface or control and encapsulate thread of control

(internal unit state, modular unit behaviour and internal rules for unit invocation). Since it's autonomous, only data crosses agents' boundaries. The Action model depends on the context and requires a balance between environment perception and representation to represent the world. A set of agents is called Multi-Agent System.

Agent Behaviour Agent has 4 main features:

- Reaction. Ability to respond to changes in time based on present.
- Situatedness. Model of action coupled with the context/environment which interacts with.
- Proactiveness. Generative approach, agents generate their objectives, encapsulate control, take the initiative.
- Social Ability. Ability to interact with other agents via some kind of agent communication means.

2.3 Self-organised Systems (SOS)

MAS, since it can model and engineer complex systems, is a SOS. In fact, it has agents for autonomy and encapsulation of behaviour, situated agents for local actions and perceptions, distribution of components and environment that supports interaction through coordination and support for organisation and cooperation concepts that are SOS requirements.

The coordination of tasks and constructions' regulation are not dependent from the workers but from constructions (stimulation stigmergy). Coordination models are the core of complex MAS.

Stimulation stigmercy It's a set of coordination mechanisms mediated by the environment. Examples of stimulation stigmercy are ant colonies that leave chemical substances marker for specific activities (pheromone). This practice has been applied to industries, i.e. automated vehicle guided.

Self-organisation

Process in which pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the system. The rules specifying interactions among the systems components are executed using only local info, without reference to the global pattern. Main elements are: increasing order due to increasing organisation, dinamism, autonomy and adaptiveness. Self-organisation is the source of emergence.

Emergence It's the arising of novel and coherent structures, patterns and properties during the process of self-organisation in complex systems. The properties are: novelty, unpredictability from low-level components; coherence, a sense of identity maintained over time; macro-level, happens at an high level; dynamism, arises over time, not pre-given.

2.3.1 Nature-inspired Computing

Observe and model complex physical systems (laws of interaction taken as given) allows to design and build complex computational systems (define laws of interaction). Stimulation stigmercy is an example as nature can provide approaches to solve complex systems issues. In fact, computational systems take the cue from nature and determine nature-inspired computing (NIC) models. In this scenario the environment is essential as a mediator for component interaction. The main NIC issues are: how to design and engineer MAS environment and how to embed stochastic behaviours in a MAS (non-deterministic nature) representing the interactions that characterize this kind of systems.

2.4 Cooperation for complex systems

Cooperative work is an interdependence of activities that occur thanks to systems interaction: an element action will change the state of a set of objects and processes with implication for the work of other ensemble members in the common field of work. The cooperative work can be distributed and involve multiple elements in interdepent activities. This is the reason why cooperative effort must be coordinated, aligned, integrated and meshed to prevent chaos.

Colloperation can be separated into 3 different branches:

- Coordination. Individuals coordinate actions each other, concurring to its completion. Main issue: synchronization. Break down the process into appropriate component actions;
- Collaboration. Individuals work together to produce a results of all individual contributions and share knowledge. Success depends on having a common goal and process understanding;
- Co-decision. Individuals collect all information needed to take a joint decision evaluating pros and cons.

Coordination is the glue that binds seprate activities into an ensemble.

2.4.1 Coordination

Coordination provides a framework in which the interaction agents can be expressed, it should cover the issues of creation and destruction of agents, communication among agents, and spatial distribution of agents, as well as synchronization and distribution of their actions over time.

Coordination model is used to:

- provide high-level abstractions and powerful mechanisms for distributed system engineering;
- enable and promote the construction of open, distributed, heterogeneous systems;
- intrinsically add properties to systems independently of components.

The main components of a coordination model are:

- Coordination entities. Entities whose mutual interaction is ruled by the model, also called the coordinables;
- Coordination media. Abstractions enabling and ruling interaction among coordinables;
- Coordination laws. Laws ruling the observable behaviour of coordination media and coordinables, and their interaction as well;
- Coordinables. Entity types coordinated, i.e processes, threads, concurrent objects and even users with an observable behaviour;
- Coordination Media. Media making communication possible among the agents, i.e. semaphores, channels, blackboards, pipelines.

Coordination Laws

A coordination model should dictate a number of laws to describe how agents coordinate themselves through the given coordination media and using a number of coordination primitives. Examples are laws that enact either synchronous or asynchronous behaviors or exploit explicit or implicit naming schemes for coordination entities. Coordination laws rule the observable behaviour of coordination media and coordinables, as well as their interaction.

2.5 Computer Supported Cooperative Work

The understanding of nature and characteristics of cooperative work is an identifiable research field called Computer Supported Cooperative Work (CSCW). This activity has the objective of designing adeguate computer based technologies to support such cooperative work. It supports men managing the complexity and providing new ways of using traditional tools or new tools to improve organization management.

Organizations do not run on information, they run on relationships. They should invest in communication and knowledge management in order to serve customers by satisfying their needs trough cooperative work. Developments in the field of CSCW might become the glue for the traditionally separate aspects of computing in work settings: single-user applications (personal computer) and organization-wide systems (transaction processing).

2.5.1 Facilities for supporting CSCW

Schmidt and Simone in [19] define 2 different strategies to support cooperative work:

- Regulating interaction: providing normative models of cooperation to regulate routine coordinative activities and enable cooperative ensembles to perform more reliably and efficiently.
- Mediating interaction: offering a medium such as "shared work space" through which entities can interact directly. According to some coordination laws enacted by the behaviour of the medium that define the semantics of coordination. (Medium coordination)

Regulating interaction

It's a coordinative interaction restricted to a priori model of interactions. It's possible to combine regulation with flexibility in 2 different ways:

- Looking for a language that incorporates an onotology of cooperation, i.e. language/action perspective, led to development of systems such as action wf.

- Selecting a suitable metaphor for modelling cooperative work. Metaphors privilege a type of wf and takes other types as subordinated and derivated.

Both of the two approaches are focused on only one aspect of coordination and couldn't be integrated with technologies supporting other aspects. This problem generated increasing interests in looking for alternative approach. Indeed, novel forms of cooperation were developed where coordination has inherently emergent behaviour property, i.e. network communities.

Mediating interaction

It's a fully unstructured approach based on ad hoc improvisation, uniqueness of events and situations. The articulation of activities counts on reliance of mutual awareness. The infrastructure provides basis for application but leaves users manage coordination complexity. There are two main kind of frameworks that are used:

- media space: space as combination of a virtual space and physical environment equipped with open audio/video channels to promote mutual awareness

- shared object space: virtual space, populated by object with cooperation capabilities supported by a suitable architecture managing the lower level services

2.6 Gap in CSCW

Schmidt and Simone identify a gap in CSCW that is showed here: "On one hand a strategy which aims at the provision of systems that prescribe interaction but which, in doing so, restricts coordinative interaction to a priori models of interaction or to a set of narrow and incongruent metaphors of interaction. On the other hand a position which aims at radically flexible systems that do not prescribe interaction but rather provide an infrastructure which can provide the basis for any application but which, in doing so, leaves it to the users to cope with the complexity of coordinating their cooperative activities. The problem is that cooperative work in the real world is not conducted according to the division of labor of CSCW research. In cooperative work, ad hoc alignment and improvisation on the basis of mutual awareness is inexorably interlaced with the execution of predefined procedures and similar formal constructs, and vice versa."

2.7 Mutual awareness

It's an elementary or undifferentiated consciousness of or sentience to the state of the cooperative effort. Consider a given cooperative work arrangement:

- When an actor changes the state of the common field of work, this will emit signs, cues, signals of this change which the other actors may perceive. Actors can perceive the state of the field of work in its entirely or a selection of it.
- From the previous one, members can develop awareness of the activities undertaken by colleagues and infer the plans/intentions of the colleagues.
- If the members of the ensemble share the same space, they also perceive each other bodily conduct, i.e. air traffic controller overhears radio conversation and takes appropriate steps
- Articulation work is a recursive phenomenon. Activities through which actors align their activities with respect to those of their colleagues can be perceived in the same way and become an additional source of awareness.

Provision of information Achieved in the course of doing the work (emission/dissemination of signals,..)

Aquisition of information It's non-intrusive. It doesn't itervene in the flow of work by enforcing the response.

Mutual awareness production

Mutual awareness is produced through very delicate practices:

- Actor modulates their activities to provide their colleagues with wfs pertinent to their being aware of these activities
- Actors make their own work visible in a form and at a level of granularity which is relevant to the situation facing the colleagues. The message is tailored to particular audience.
- Actors continually monitor or scan activities of their coleagues to ascertain the state of these activities.

2.8 Coordinative artifacts and protocols

Cooperative work is constituted by interdependence of activities. It is so complex that the articulation of cooperative work in terms of ad hoc interaction and improvisation based on mutual awareness is quite insufficient. This is the reason why new modalities are required, in particular the use of artifacts for coordination purposes and implicit/explicit protocols to regulate the artifacts use. Coordinative artifacts are used in different ways and for different cooperative work purposes:

- maps, i.e. artifacts that specify interdependencies of tasks or objects in a cooperative work setting
- templates, i.e. artifacts that specify properties of the results of individual contributions
- scripts, i.e. artifacts that specify a protocol of interaction in view of tasks interdependencies in a cooperative work setting

Coordinative artifacts stand proxy for the affordances and constraints of the environment.

2.8.1 Protocol

It's an integrated set of procedures/conventions which stipulate the articulation of distributed interdependent activities. It provides a "precomputation" of activities which actors can rely on to reduce the space of possibilities (articulating cooperating work). It's a linguistic construction that can't deal with "the vagueness of plans", the detail of intent and action must be contingent on the circumstantial and interactional particulars of actual situations. It's entirely decontextualized, but deliberately under-specified with respect to factors immaterial for the purpose of the given protocol and those that can be left unspecified. Coordinative artifact will encounter situations beyond its bounds, inherent vagueness and appropriate ambiguity, "no representation of the world is either complete of permanent".

2.8.2 Artifact

A distinct and permanent symbolic construct. According to the flow oriented-approach it's complementary to the flow as part of the context where flow takes meaning and supports info delivery. An alternative approach suggests that supports application development and is assisted by technology infrastructures evolution providing integration of different communication media, easier interface design, distributing application across heterogeneous machines and open infrastructure across www.

Protocol is objectified in the artifact that gives a performance to the protocol. Artifact is a representation of the state of the execution of the protocol.

Objective artifact (environment central element), subjective artifact (agent and its autonomy as central element).

Elemental categories

Protocols elemental categories capture how actors identify the basic units of work and information for articulating their activities and the relation between these units. All entities must be able to communicate to ensure the different intrusive modes of articulation work. Any type of flow of work must be representable and it's not imposed a specific modelling approach.

For mutual awareness capture how actors identify which basic units of information have to be associated to which basic types of triggers of info pertinent to mutual awareness formations. All entities must be able to communicate to ensure different levels of intrusiveness with respect to the recipient's course of action.

2.9 Articulation work

Articulation work ss characterized by a continuum of interactional modalities from improvised ad hoc iteraction and adoption of learned common-sense constructs to the design, introduction and application of coordinative artifacts and protocols. On one hand, actors cooperating in shared space of artifacts are engaged in articulating distributed activities. On the other hand, actors routinely rely on the common-sense, artifacts and actors as competent members. With highly complex interdependencies, actors rely on implicit conventions but will devise well designed coordinative artifacts and protocols.

Protocols execution presumes and involves situated action, that requires actors' mutual awareness with respect to the state of the field of work, of the cooperating ensemble, of the protocol itself and of intersecting protocols. Information about provision and aquisition of mutual awareness is a primary resource for an effective and adaptable use of protocols to support distributed activities.

Combined use of coordinative protocols and mutual awareness allows to reduce computational cost through precomputation without imposing rigidity to the flow of activities, and to take the best decision knowing the current state of affairs. There are 3 main consequences:

- mutual awareness fundamental for articulation work. It provides a basis for ad hoc improvisation in shared spaces and rich texture which enables actors to specify and situate protocols.
- Technology that supports cooperative work can't avoid to adress protocols and mutual awarenss in an integrated way using the best mean of coordination.
- Protocols and awareness functionality, adaptability by the users to their current situation is a mandatory requirement.

Dynamic Systems Requirements Flexibility and adaptability of the technology in relation to work settings are characterized by dynamic set of requirements. The first approach to deal with this peculiarity is to define a common space where people access and manipulate shared artifacts and work process is not codified because it can change at any time. Flexibility is responsibility of articulation work on users and functionalities can be part of a CSCW infrastructure. A second approach defines features and languages to express formal constructs to supper adaptability at the semantic level of articulation of work.

2.10 Situatedness

It's the property of systems of being immersed in their environment. That is, of being capable to perceive and produce environment change, by suitably dealing with environment events. Essentially, it concerns interaction between processes and the environment. Technically, it can be conceived as a coordination problem how to handle and govern interaction between pro-active processes and an ever-changing environment.

Due to the increasingly complex requirements for our computational systems there's an ever-increasing context-awareness. Defining the notion of context

is a complex matter, with its boundaries typically blurred with the notion of environment.

Mobile computing scenarios have made clear that situatedness of computational systems nowadays requires at least awareness of the spatio-temporal fabric. That is, any non-trivial system needs to know where it is working, and when, in order to effectively perform its function. In its most general acceptation, then, any environment for a computational systems is first of all made of space and time. A spatial event should be generated within a coordination medium, conceptually corresponding to changes in space. Then, such events should be captured by the coordination medium, and used to activate space-aware coordination laws. Spatial coordination requires spatial situatedness and spatial awareness of the coordination media, whereas time-dependent coordination policies need a time-aware coordination medium.

Chapter 3

Cooperative work implementation

In the previous section, concepts that are at the basis of coordination computer-based facilities for supporting cooperative work are being introduced. Now, I'm going to delineate technologies that are adopted to represent previous concepts. In particular, I'm going to outline tuple-space technology and an integrated support of articulation work (AWR approach: Ariadne, AW-Manager, Reconciler) constituted by: Ariadne, a language for coordination mechanisms (coordinative protocols and artifacts), AW-Manager, a software component that collects a set of linguistic features to define mutual awareness between actors and artificial entities, and Reconciler, a software component that supports the interoperability of different "coordination mechanisms".

3.1 Tuple-space meta-model

This technology is made by tuples, an ordered collection of knowledge chunks, possibly heterogeneous information. Coordinables synchronise, cooperate, compete based on tuples available in the tuple space by associatively accessing, consuming and producing tuples. The main features are:

- Coordination media: tuple spaces as multiset / bag of data objects / structures called tuples;
- Generative communication: until explicitly withdrawn, the tuples generated by coordinables have an independent existence in the tuple space; a tuple is equally accessible to all the coordinables, but is bound to none;
- Communication language: tuples as ordered collections of (possibly heterogeneous) information items. (i.e. Linda, set of templates specifications of set/classes of tuples. It has a mechanism that matches tuples and templates.);
- Coordination language: tuple space primitives as a set of operations to put, browse and retrieve tuples to/from the space;

- Associative access: tuples in the tuple space are accessed through their content and structure, rather than by name, address, or location;
- Suspensive semantics: operations may be suspended based on unavailability of matching tuples, and be woken up when such tuples become available.

3.1.1 Coordination Problems

Coordination is limited to writing, reading, consuming, suspending on one tuple at a time. The behaviour of the coordination medium is fixed once and for all. Coordination problems that fits it are solved satisfactorily, those that do not fit are not. As a result, the coordination load is typically charged upon coordination entities. This does not fit open scenarios neither it does follow basic software engineering principles, like encapsulation and locality.

Coordination Solutions

Making the behaviour of the coordination medium adjustable according to the coordination problem and the changing environment could be a solution. If the behaviour of the coordination medium is not be fixed once and for all, it can be defined in accordance to the coordination needs. Then, in principle all coordination problems may fit some admissible behaviour of the coordination medium with no need to either add new ad hoc primitives, or change the semantics of the old ones. In this way, coordination media could encapsulate solutions to coordination problems represented in terms of coordination policies, enacted in terms of coordinative behaviour of the coordination media. What is needed is a way to define the behaviour of a coordination medium according to the specific coordination issues: a general computational model for coordination media, along with a suitably expressive programming language to define the behaviour of coordination media.

According to this analysis, new features have to be determined that allow to:

- define new coordinative behaviours embodying required coordination policies
- associate coordinative behaviours to coordination events

Keeping information representation and perception separated in the tuple space would enable process interaction protocols to be organised around the desired / required process perception of the interaction space (tuple space), independently of its actual representation in terms of tuples. Processes could get rid of the unnecessary burden of coordination by embedding coordination laws into the coordination media.

A solution could be to maintain the standard tuple space interface and make it possible to enrich the behaviour of a tuple space in terms of the state transitions performed in response to the occurrence of standard communication events. So, in principle, the new tuple-based abstraction should be a tuple space whose behaviour in response to communication events is no longer fixed once and for all by the coordination model, but can be defined according to the required coordination policies. Another solution could be the Tuple Centre.

Another possible solution: Tuple Centre Since it has exactly the same interface, a tuple centre is perceived by processes as a standard tuple space. However, since its behaviour can be specified so as to encapsulate the coordination rules governing process interaction, a tuple centre may behave in a completely different way with respect to a tuple space.

A tuple centre is a tuple space enhanced with a behaviour specification, defining the behaviour of a tuple centre in response to interaction events.

The behaviour specification of tuple centre is expressed in terms of a reaction specification language, and associates any tuple-centre event to a (possibly empty) set of computational activities, which are called reactions. More precisely, a reaction specification language enables the definitions of computational activities within a tuple centre, called reactions, and makes it possible to associate reactions to the events that occur in a tuple centre.

Tuple Centre's State vs. Process's Perception Reactions are executed in such a way that the observable behaviour of a tuple centre in response to a communication event is still perceived by processes as a single-step transition of the tuple-centre state as in the case of tuple spaces. So tuple centres are perceived as tuple spaces by processes. Unlike a standard tuple space, whose state transitions are constrained to adding, reading or deleting one single tuple, the perceived transition of a tuple centre state can be made as complex as needed.

This makes it possible to decouple the process's view of the tuple centre (perceived as a standard tuple space) from the actual state of a tuple centre, and to relate them so as to embed the coordination laws governing the distributed system.

On the one hand, a tuple centre is basically an information-driven coordination medium, which is perceived as such by processes. On the other hand, a tuple centre also features some capabilities which are typical of action-driven models, like the full observability of events, the ability to selectively react to events, the ability to implement coordination rules by manipulating the interaction space.

Social tuple centre in distributed environment Individual tuple centre handles individual behaviours and state, and mediate interaction of individuals with social tuple centre. The social tuple centre deals with shared resources and ensures global system properties, like fairness and deadlock avoidance. At any time, one could look at the coordination media, and find exactly the consistent representation of the current distributed state properly distributed, suitably encapsulated.

3.1.2 ReSpecT tuple centres for environment engineering

ReSpecT tuple centres is a language that is able to capture, react and observe general environment events. What's more, it generally interacts with the environment and mediates process-environment interaction expressing general MAS-environment interactions. ReSpecT tuple centres are:

- inspectable: both tuple space (tuples for knowledge, information, messages, communication) and space specification (tuples for behaviour, func-

tion, coordination) are inspectable by engineers.

- malleable: the behaviour of a ReSpecT tuple centre is defined by the ReSpecT tuples in the specification space and can be adapted / changed by changing its ReSpecT specification.
- linkable: all primitives could be executed within a ReSpecT reaction as either a reaction goal executed within the same tuple centre or as a link primitive invoked upon another tuple centre.

As a behaviour specification language, ReSpecT enables the definition of computations within a tuple centre—called reactions. It makes it possible to associate such reactions to events occurring in a tuple centre. ReSpecT has both a declarative and a procedural part. As a specification language, ReSpecT allows events to be declaratively associated to reactions by means of specific logic tuples, called specification tuples. By adopting the declarative interpretation, a ReSpecT tuple centre has then a twofold nature: a theory of communication – the set of the ordinary tuples – and a theory of coordination—the set of the specification tuples. This allows in principle intelligent agents to reason about the state of collaboration activities and to possibly affect their dynamics.

Source and target of a tuple centre event can be any external resource. A suitable identification scheme – both at the syntax and at the infrastructure level is introduced for environmental resources. The ReSpecT language is extended to express explicit manipulation of environmental resources.

Transducers Specific environment events have to be translated into well-formed ReSpecT tuple centre events This should be done at the infrastructure level, through a general-purpose schema that could be specialised according to the nature of any specific resource. A ReSpecT transducer is a component able to bring environment-generated events to a ReSpecT tuple centre (and back), suitably translated according to the general ReSpecT event model. Each transducer is in charge of handling the specific resource and aimed at handling, like a temperature sensor, or a heater.

TuCSoN The TuCSoN meta-coordination language allows agents to program ReSpecT tuple centres by executing meta-coordination operations. TuCSoN provides coordinables with meta-coordination primitives, allowing agents to read, write, consume ReSpecT specification tuples in tuple centres and also to synchronise on them.

Coordination middleware (tuple-based) is fundamental to build complex software environment. The more the system becomes complex the more abstractions and metaphors has to be included. This is the reason why coordination middleware implements abstractions to embed situatedness and stochastic behaviours social rules (norms, laws).

3.2 AWR: Towards integrated support of articulation work

The main concepts of an integrated support environment to support articulation of work are coordinative protocols and mutual awareness, achieved through the

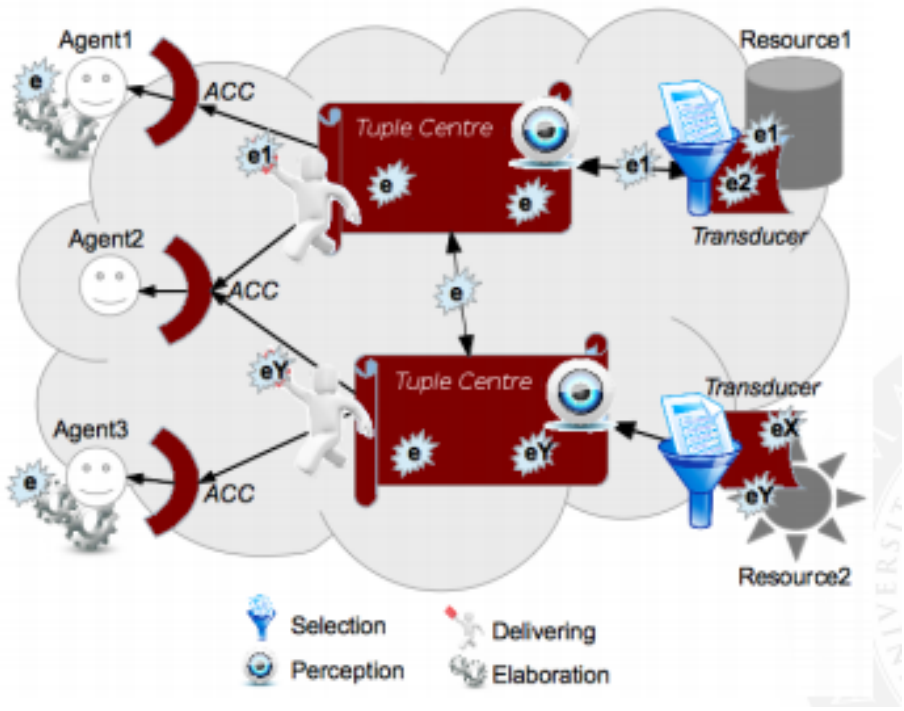

Figure 3.1: ReSpecT - TuCSoN architecture

design and implementation of a series of prototypes: Ariadne for coordinative protocols and artifacts, AW-Manager for mutual awareness and Reconciler for interoperability (AWR: Ariadne, AW-Manager, Reconciler)

3.2.1 Ariadne

It's a language for "coordination mechanisms": coordinative protocols and artifacts, e.g. standard operating procedures supported by a form. It's developed in ABACO (Agent-based architecture) and offers interoperability to support flexibility at various levels. Ariadne provides the designer with the possibility of designing different, specialized languages for describing coordination mechanisms starting from a set of basic elements. The designer can adapt the specific situation under concern. Ariadne main property is flexibility: the full composability of its constitutive elements (categories of articulation work, its relation and aggregations) obtained through definition of a small set of communication primitives allowing these components to combine their behaviour ([4]).

Key Concepts

- Cooperative work. Interdependence of multiple actors who interact through change the state of a common field of work.
- Articulation work. The need to restrain the distributed nature of complexly interdependent activities.

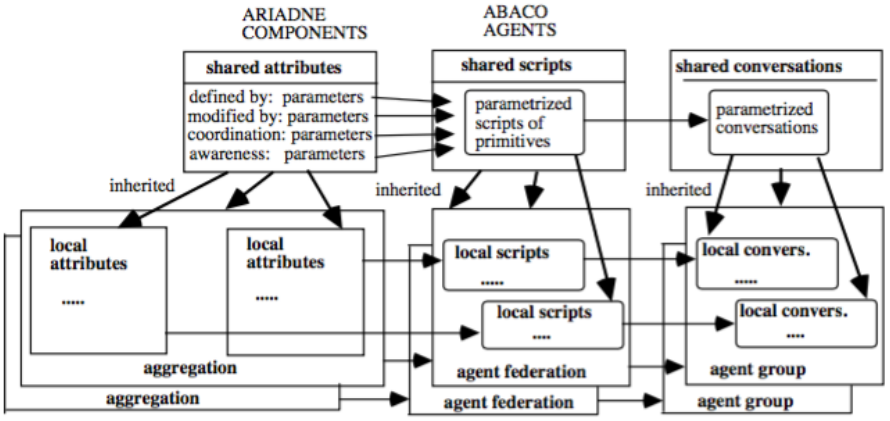

Figure 3.2: Ariadne Architecture and its implementation

- Field of work. The part of the world affected by actors' work (data structures and functionalities of the application).
- Coordination Mechanism (CM): protocol and artifacts.
- Computational Coordination Mechanism (C2M). Software device incorporates the computational artifact and the protocol.

Main Components

- Set of basic elements. Object of Articulation Work (OAW), closed space of possibility from which to choose the C2M elemental constituents.
- Set of basic relations. Combination of OAW into the protocols and artifacts that constitute the C2Ms.
- Set of primitives. Guide the users in defining, adapting, linking C2Ms.

ABACO

It's a multi-agent architecture organized into layers that mimic the conceptual structure of Ariadne. Each component of Ariadne is realized as an agent belonging to a specific type: each category corresponds to a specific agent, aggregation of categories corresponds to a federation of agents. The mapping of Ariadne elements' on ABACO gives a formal definition of Ariadne semantics and defines the structure of a crucial component of a CSCW architecture specialized to support the construction of computational coordination mechanism. ABACO allows to satisfy the two basic requirements of Ariadne: malleability and linkability. The first property allows to adapt a C2M to new demands in the course of its lifecycle whereas the second one allows to link a C2M to other C2M in a wider organizational context. The architecture needs to be modular to implements these two requirements that lead to dinamicity (agents and their interactions). ABACO use weak agency: agents constituted by hardware or software-based computer system with: autonomy, social ability, reactivity, proactiveness ([5]).

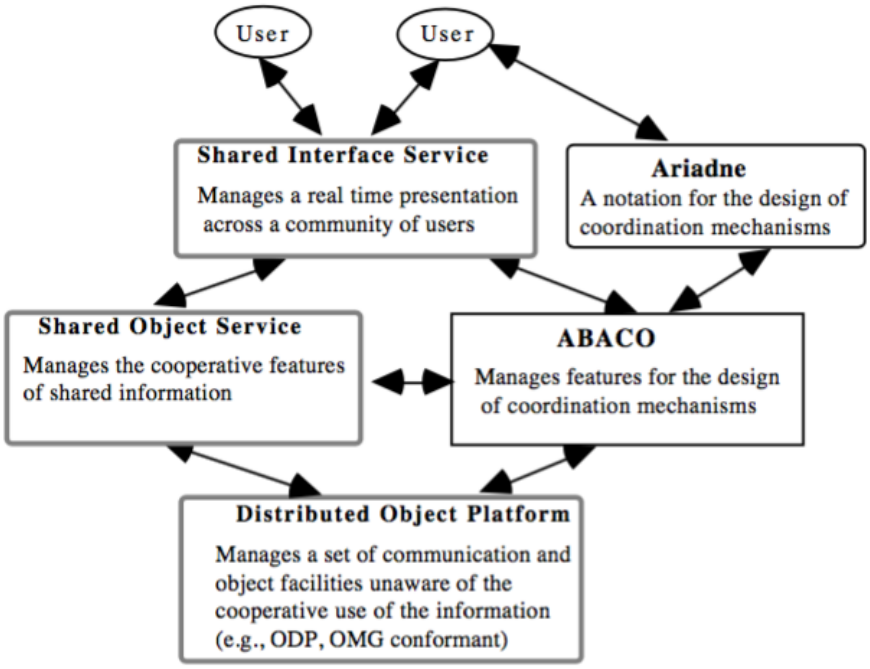

Figure 3.3: Abaco as a component of a proposed CSCW architecture

The layered structure

1. OAWs (C2M building blocks). Ariadne agents specialised to the management of the information characterising them and relationship with their environment

2. Ariadne agents capturing communication capabilities to its active artifacts and proctors. User Interface as destination of the message of user's awareness of what is going on in the field of work and the cooperative work arrangement. Agents are responsible for communicating with the users.

3. Ariadne agents obtained from the composition of already existing C2M based on the definition of an interface agent specialized to manage external communication and to support a first degreee of tolerance of the modifications implied by malleability requirement. Interface agent is needed to establish the type of interoperability between a C2M and the field of work.

3.2.2 AW-Manager

It's a software component that collects a set of linguistic features to define mutual awareness between actors and artificial entities. It has specification of triggers and allows provision, acquisition of and reaction to awareness information.

3.2.3 Reconciler

It's a software component that supports the interoperability of different "coordination mechanism" in terms of mutual alignment of their boundary objects and events.

3.3 AWR approach problems

As stated in [19], Schmidt and Simone argued that must be adopted a uniform approach to the definition of the capability supporting any of them to integrate coordinative protocols and mutual awareness. Users must adapt mutual awareness and protocol functionalities in a way that is adeguate to their needs, they must be flexible. First of all it's important to combine a language for "coordination mechanisms"(Ariadne) with a software component that collects a set of linguistic features to define mutual awareness between actors and artificial entities (AW-Manager) incorporating functionalties based on a limited number of elemental categories that can be composed flexibly to obtain more complex behaviour. We need languages that allow users to express salient categories of articulation work and rules for combining them managing coordinative protocols and mutual awareness. Secondly, categories and composition rules have been made visible to the users that can freely decide the appropriate level of adaptation.

3.3.1 Integration problem

Elemental categories and composition rules should reflect distinctions preserving integrability of coordinative protocols and mutual awareness functionalities. A solution could be to provide each category within the Ariadne language with mutual awareness capabilities in such a way mutual awareness can be "expressed" at any level of depth in protocols definition. Otherwise, another approach is the following:

1. AW-Manager is notified by categories constituing the protocol of the awareness triggers using their communication capabilities
2. AW-Manager handles the awareness space and scripts execution (triggers associated)
3. AW-Manager returns awareness information to the appropriate categories

So, Coordinative artifacts in articulation work conveying awareness information of current state to actors and computational artifacts are able to acquire awareness and adjust their activities achieving an active role and flexibility.

3.3.2 Distributedness problem

As well as cooperative work also coordinative protocols and mutual awareness integration have distributed nature. Due to its nature, cooperative applications raise a problem of mutual alignment. This is the reason why interoperability has to be introduced at the semantic level of articulation work. To deal with this issue we need a rich contextual information, in terms of modifications, current states of the involved protocols and conventions governing the definition

and use of boundary objects with the related events. In order to supply this kind of subject two software components are used: contextual info service and the reconciler. The reconciler tries to keep mutually interpretable the interacting coordination mechanisms associated with entities that populate a set of interacting awareness space. This space is managed by AW-Manager defined by actors that are dynamic by nature and referred to awareness space and the concept of dissemination that is employed in mutual awareness.

Chapter 4

Workflow and CSCW theories

As well as most important fields in research activity, it's important to understand the path that research has taken and where it's going. So, I tried to focus on CSCW and workflow basic thoeries and concepts. Workflow system is a CSCW technology that supports the governing of divided work processes. On one hand CSCW has a low level of coordination and an high level of possibility in interaction, on the other hand WFMS has an high level of coordination and a low level of possibility in interaction.

4.1 CSCW characterization

In 1996 Nutt created a 3-dimension CSCW characterization that could help to describe the alternative forms of CSCW giving an informal characterization of different CSCW system approaches ([15]).

The x-axis

"It informally represents the amount of logical immersion a human worker has in the system. " The first level of immersion into the work can be supported using the telephone, the environment among the collaborators is loose and informal, whereas the last level of immersion into the work is the virtualization of a shared room and its artifacts, including its participants.

The y-axis

"It describes the amount of computation support the system provides human users." The level of domain-specific computation support is an important criterion in characterizing a CSCW system. Thus the first level of computation support is the presence of generic, high level programming languages for the system whereas the final level of computation support is computation support through system facilities for contextual assistance.

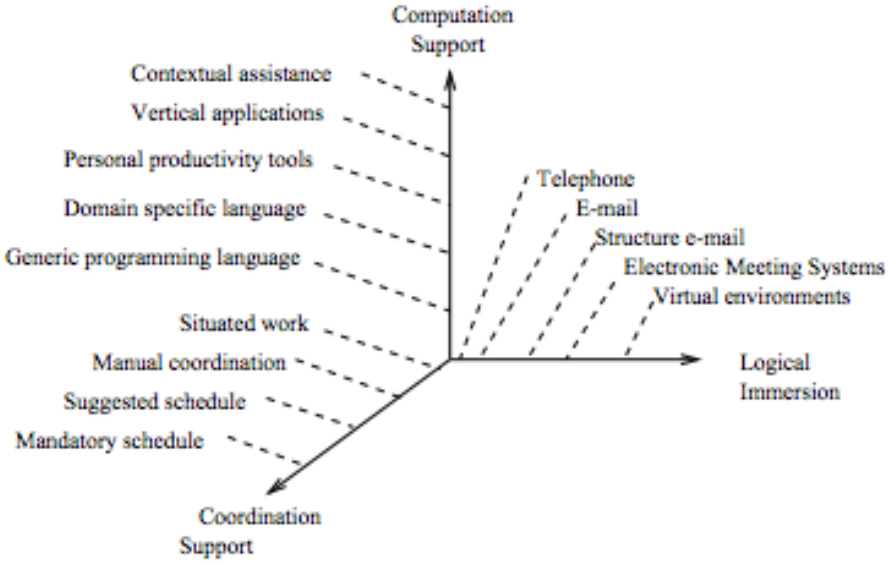

Figure 4.1: CSCW characterization

The z-axis

"The z-axis represents the amount of coordination support the system provides." CSCW systems embracing the situated work philosophy attempt to design the system so it creates an electronic environment that is perceived to be similar to (or at least only incrementally different from) an extant manual environment. Situated work tends to minimize the importance of automatic coordination. Manual coordination refers to systems providing tools to enable the group to coordinate their activities, including shared memory, shared resources, and synchronization mechanisms such as locks. In contrast to the situated work proponents, researches argue that computers should explicitly be used to coordinate tasks. Workflow models are one approach for doing this, though there are other models for accomplishing this task.

4.2 Two ways to support cooperative work

Nutt, in its article [15] argued that a flexible CSCW system should allow its designers and users to control the point along the axis at which roles are executed. This provides the primary justification for incorporating workflow models into a virtual environment. In fact, the presence of the procedure model provides the environment's users with an artifact to represent and discuss the work they are trying to accomplish.

The models and systems must support varying degrees of conformance to the coordination model specification, depending on the situation. The goal of leading edge systems is to support these extended workflow enactment languages in a collaboration environment supporting high logical immersion into the en-

vironment, with high computation support, and variable coordination support.

4.2.1 Situated Work

The situated work camp advocates the use of evolutionary systems to provide increasingly sophisticated personal productivity tools. The human participants decide how tools should be applied to perform the work, and how work should be routed from one person to another. Plans are representations of situated actions produced in the course of action and therefore they become resources for the work rather than they in any strong sense determine its course.

4.2.2 The workflow

The workflow camp advocates the use of models and systems to define the way the organization performs work. In particular, a work procedure in a workflow system is defined by a workflow model composed of a set of discrete work steps with explicit specifications of how a unit of work flows among the different steps.

Whereas the situated work approach explicitly omits step definition and inter-step coordination, workflow explicitly includes it. Besides assisting in the coordination and execution of a procedure, workflow models have also been used to describe how the procedure performs the work, and to analyse the behaviour of a procedure.

This leads us to the idea that the contribution of computer technology might best be achieved by taking a perspective in which the system can be used to support both situated work and workflow (or some customized combination of the two) on a case-by-case basis. Therefore, in workflow technology, we see an evolution towards flexible workflow models and systems that provide robust means for supporting situated work in a distributed environment.

Workflow system flexibility fundamentals

A flexible language is a necessary but not sufficient part of an effective workflow system. Workflow enactment systems, that support the execution of workflow processes, must also be broadened from their current form to take advantage of these language extensions. They must also provide increased flexibility in the style of collaborative work they support.

The workflow system must also explicitly address the general characteristics of a collaborative system, otherwise it will ignore the general issues we have raised. By understanding those issues, it is possible to design a broader class of systems that meet the needs of the team of workers.

4.3 Workflow criticism

Over the years workflows had been subject to much controversy and criticism for their rigid representation of work in process models. The potential danger with current workflow systems was that their design is predictated entirely by formal procedures ignoring (and even damaging) the informal practice being too prescriptive and rigid. So, Researchers began to study new theories that could rethink or replace workflow systems: Activity theory and the Spatial Model.

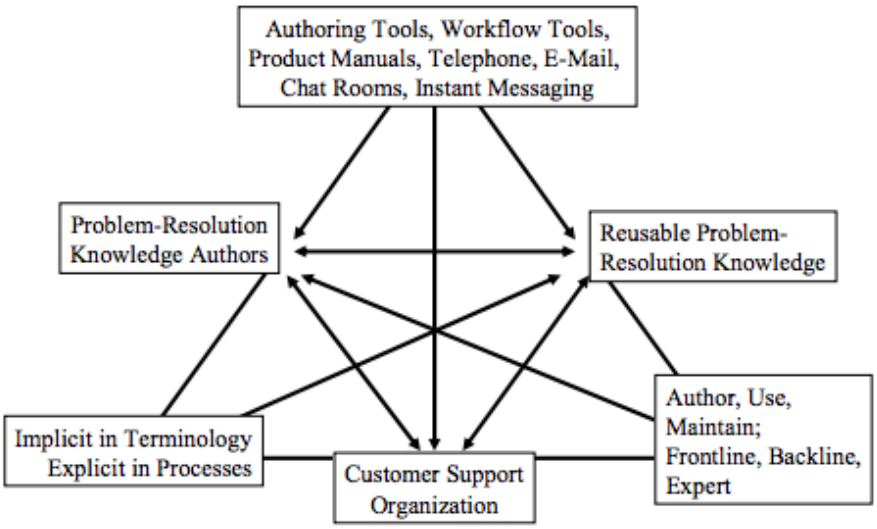

Figure 4.2: Activity System Model

4.4 The Activity Theory

Activity Theory originated in the former Soviet Union as part of the cultural-historical school of psychology founded by Vygotskij, Leontjev and Lurija. The theory is a philosophical framework for studying different forms of human praxis as developmental processes, with both the individual and social level interlinked.

The fundamental unit of analysis is the human activity which has three basic characteristics:

- it is directed towards a material or ideal object which distinguishes one activity from another;
- it is mediated by artifacts (tools, language, etc.);
- it is social within a culture.

In this way, computer artifacts, like all other artifacts, mediate human activity within a practice. By acting in the world, human beings meet the objective world, which is experienced through the activity. Thus, human knowledge about the world is reflection obtained through activity, constituting the basis for expectations, and desires about activities in this world. In this framework plans guide technology and human activity role.

Activity Theory pros This theory allows to:

- identify the stakeholders in the process;
- ensure that technology is designed to benefit the user;
- work toward alignment between users' rewards and business needs;
- work toward alignment between the rewards of the designers of the device and the business needs.

4.4.1 A way to rethink Workflow as a situated planning

In 1996 Bardram ([2]) claimed that based on Activity Theory a plan can be defined as a cognitive or material artifact which supports the anticipatory reflection of future goals for actions, based on experience about recurrent structures in life.

As an artifact, the plan is socially constructed, shared among the actors in the work practice, used to mediate work, and constitute a central part of the organisation material conditions for work. A plan is a series of expectations to future results under certain conditions and the execution becomes an afferent synthesis between the plan and the conditions of the concrete situation.

The fundamental feedback loop in the course of an activity forms the basis for a learning process embedded in the activity. This learning process creates and enhances the plan, which was originally the guiding principle for the activity.

Workflow Accountability

The core point is to recognise the function of plans as ways of anticipating and pre-handling events in (working) life based on their recurrent nature, and be able to save and later reuse the experience obtained in handling these events. This understanding of plans ascentral assets in work has some implication for the issue of workflow systems: instead of supporting routing information around in organisations according to a workflow process model, the computer should be a tool mediating the anticipatory reflection of recurrent events in working life.

Based on this conceptualisation it becomes possible to make a planning tool that does not emphasise a rigid match between process models and work. However, it is central to understand why such formal process models are made and embedded in workflow systems in the first place.

Thus, workflow systems are conceived as organisational infrastructure used and designed for meeting organisational goals (e.g. customer satisfaction). When viewed from this overall organisational perspective, workflow systems are often used to keep track of the work according to these organisational goals. This means that a workflow system is not just mediating the workflow (which has been the premise for this paper so far), but is used for additional managerial purposes. Hence, the workflow system becomes a technology of accountability as defined by Suchman [21]: "By technologies of accountability I mean systems aimed at the inscription and documentation of actions to which parties are accountable [...] in the sense represented by the bookkeeper s ledger, the record of accounts paid and thosestill outstanding" (p. 188).

4.5 The spatial model

Spatial Model largely developed in the European Communities' COMIC project (1992-1995) (Computational Mechanisms of Interaction in Cooperative Work). It supposes that objects (which might represent people, information or other computer artefacts) can be regarded as situated and manipulable in some space. In principle, any application where objects can be regarded as distributed along dimensions such that their position and orientation can be measurably determined is amenable to analysis in terms of the Spatial Model.
Spatial approaches to collaborative systems became increasingly popular. The

main reason was their strong relation to physical reality and therefore their highly intuitive nature. Its model provided generic techniques for managing interactions between various objects in such environments including humans and computer artifacts. Furthermore, the model was intended to be sufficiently flexible to apply to any system. The interaction between objects in space was mediated through the relationships obtaining between three subspaces: aura, focus and nimbus.

Medium Any interaction between objects occurs through some medium. A medium might represent a typical communication medium (e.g. audio, visual or text) or perhaps some other kind of object specific interface. Each object might be capable of interacting through a combination of media/interfaces and objects may negotiate compatible media whenever they meet.

Aura It is defined to be a sub-space which effectively bounds the presence of an object within a given medium and which acts as an enabler of potential interaction. It is assumed that an object will carry with it an aura which, when it sufficiently intersects with the aura of another object, will make it possible for interaction between the objects to take place. On this view, an aura intersection is the pre-condition of further interaction.

Nimbus and Focus The subspaces of focus and nimbus are intended as representing the spatial extent of an object's 'attention' and its 'presence'. Thus, "if you are an object in space, a simple formulation might be: the more an object is within your focus, the more aware you are of it; the more an object is within your nimbus, the more aware it is of you." and accordingly, "given that interaction has first been enabled through aura collision: The level of awareness that object A has of object B in medium M is some function of A's focus in M and B's nimbus in M." ([2]).

To summarise, spatial model defines key concepts for allowing objects to establish and subsequently control interactions. Aura is used to establish the potential for interaction across a given medium. Focus and nimbus are then used to negotiate the mutual and possibly non-symmetrical levels of awareness between two objects which in turn drives the behaviour of the interactions.

4.5.1 Awareness in the spatial model

As stated in [6] the topic of awareness began to receive a great deal of attention in CSCW. Providing participants with mutual awareness of each other's activities is often seen as a important research and design goal. This research provides an alternative way of supporting cooperative work. In workflow systems work activities are given a formal representation in terms of some workflow model which often stipulates how the contributions of different participants are to be coordinated.

In contrast, in many awareness-oriented systems, the coordination between different activities is supported by giving participants an awareness of what each other are doing or have done so that participants can coordinate their work themselves. Many researchers thought that this provided more flexible

applications which are not liable to the usability criticisms that can be made of more procedural-oriented approaches to CSCW.

CSCW awareness issues

Steve Benford and Lennart Fahlen in [20] argued that there were 2 main issues about this theory. First, there exist attempts to integrate support for awareness at fundamental levels of cooperative system architecture. Secondly, some of the concepts, models and notations elaborated in the development of awareness oriented applications and systems will be found to have a broader utility.

In [6], J Hughes presents how they used the previous model and the concepts: they described the implementation of an awareness engine, called AETHER, based on the suggested model. His goal was to recognise awareness at a fundamental system level and to build other functions on top of it.

4.5.2 Artifacts autonomy in virtual worlds

Researchers thought that existing floor workflow modelling techniques were in contrast with a "spatial" approach where people employ the affordances of virtual computer space as a means of control. In so doing, Benford and Fahlen underlying philosophy had been to encourage individual autonomy of action, freedom to communicate and minimal hard-wired computer constraints. Where the interacting objects are artifacts, the model provides mechanisms for constructing highly reactive environments where objects dynamically react to the presence of others (e.g. you may activate a tool simply by approaching it).

They had chosen to base their work around the metaphor of interaction within virtual worlds. Under this metaphor, a computer system can be viewed as a set of spaces through which people move, interacting with each other and with various objects which they find there.

Virtual spaces Virtual spaces can be created in any system in which position and direction, and hence distance, can be measured. Virtual spaces might have any number of dimensions. The objects inhabiting virtual spaces might represent people and also other artifacts (e.g. tools and documents) that ensure individual autonomy.

4.5.3 Spatially-oriented system evolution

The spatial model, as its name suggests, uses the properties of space as the basis for mediating interaction. Thus, objects can navigate space in order to form dynamic sub-groups and manage conversations within these sub-groups. However, Benford and Fahlen believed that current spatially-oriented systems would not effectively scale to heavily populated spaces. More specifically, as the number of occupants in a virtual space increases beyond a few people, the need to effectively manage interactions will become critical. In fact, they thought that new techniques are needed which support natural social conventions for managing interactions. One approach might be to take advantage of the highly fluid and dynamic nature of space and combine the basic functionality discussed before to create a flexible approach for workflow support.

Chapter 5

Towards Workflow Autonomy

The main part of researchers think that workflow evolution goes through the investigation of complex physical systems. So, as well as stated in the first chapter, living systems are autopoietic units that sustain themselves on a inner network of reactions and continue to mantain the organised bounded structure (self-determined flexibility) that is an element of system identity (i.e. membrane cell), workfow systems have to be considered as a complex structure constituted by mutual interactions of its components, specifying their own rules of behaviour. This approach is strongly related to autonomy, in fact, workflow components have to be autonomous entities. The way this system takes place is very sophisticated and there are a lot of ideas and methods that are thought to set up this architecture. Stimulation stigmercy is an example as nature can provide approaches to solve complex systems issues ([17]). Information provided in the first and second chapter of this paper should allow to understand and discover new approaches that deal with complex systems issues and are described in this chapter. First of all, in this chapter I'm going to describe how elements defined in the first chapter are involved in workflow management: coordinative artifacts, TuCSoN, ReSpecT and Ariadne. Then, some theories about workflow organization extracted from the real world are introduced. Finally, I illustrate techniques that describe how self-configuration takes place in workflow management and adapts to shape workflow management autonomy.

5.1 Coordination artifacts for Workflow Management

Workflow management concerns the automated integration and coordination of heterogeneous and independent activities involved in the same global business process. Among the others it includes activity scheduling and synchronisation, information and control flow management, exception management, and so on. Currently, in the context of service-oriented architectures – in particular Web Services – workflow management is also called orchestration.

The dynamic support for balancing task automation with cooperation flex-

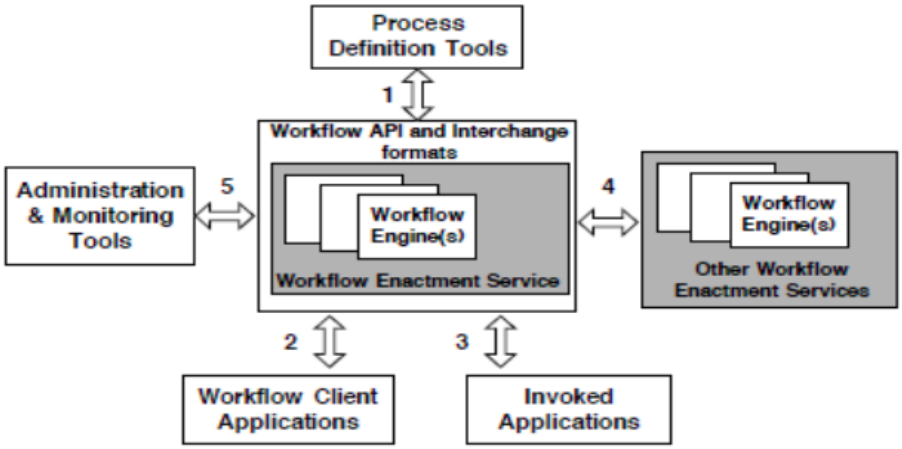

Figure 5.1: WFMS definition

ibly is among the most important requirements for state-of-the-art systems for workflow management, supply chain management, and CSCW. The ability to change the "engineering point" of coordination dynamically is specially required in open systems, where the environment can unpredictably change, the goals of the system can evolve, and coordination laws can be improved as a result of agent interaction.

5.1.1 TuCSoN and ReSpecT flexibility

In the case of TuCSoN, this is achieved by means of the ReSpecT tuple centre model, and the tools provided by the infrastructure. The coordination laws that define the behaviour of the coordination media (tuple centres) expressed in the ReSpecT language can be inspected and changed dynamically by human and artificial agents by means of specific tools. The evidence of the effectiveness of this approach can be observed inter-organisational workflow management systems where: tuple centres have been used to embody the role of workflow engines, and workflow rules have been described as ReSpecT coordination laws. Each workflow engine (tuple centre) acts as a coordination artifact providing fluid coordination of the individual tasks executed autonomously by human and artificial agents.

So, it's possible to identified 2 stages whereby workflow rules are:

- inspectable by inspecting the coordination laws embedded in tuple centres (reflection stage);
- modifiable at runtime by changing the coordination laws of tuple centres (reification stage).

5.1.2 Cooperation requirements needed

Distributed workflow management systems (WfMS) and office automation environments require the automatism and prescriptiveness. Unstructured environ-

ments (from the coordination point of view) and market-oriented competitive scenarios mostly rely on the reasoning and interaction capabilities of individual agents.

Moreover, relevant coordination scenarios, such as business process coordination and CSCW, require models providing the means for balancing subjectivity and objectivity, mediating between application-centric and human-centric tasks, unstructured process and highly structure process structure. In order to face such a complexity, we consider useful from both a scientific and an engineering point of view to provide a conceptual framework exploiting both the subjective and the objective coordination approaches: this is important to ground models and infrastructures aiming at supporting both approaches in MAS coordination.

5.2 Wfms adaptability with Ariadne

Workflow system adaptability can be effectively reached if different dimensions of process modeling are involved. As stated in [4]: "These dimensions concern the possibility to flexibly combine a rich set of basic categories in order to obtain the most suitable language to model the target business process and the work practices around it; to take into account various levels of visibility of the process model in relation to the needs and the role of the actors who utilize it during its life cycle; and finally, to allow for temporary as well as permanent modifications of the process itself."

Ariadne role In this scenario, Ariadne realizes adaptability in the design of a technological support of workflow management, starting from the established work practices and following their evolution. Moreover, as stressed in the first chapter, the role of awareness promotion is also fundamental in the modifications process. This opens one of the most challenging issues in process modeling: namely, the smooth integration of features and tools supporting awareness not only with features and tools supporting the use of formal constructs describing more recurrent patterns of cooperation (as in Ariadne) but also, and especially, with those features and tools which are devoted to support modifications. This challenge has to take into account the distributed nature of cooperation as explicitly required by the more recent evolutions of the organizations hosting the workflow systems, among others, virtual and networked organizations and various forms of electronic commerce.
Hence, adaptability in a distributed environment raises the problem of workflow interoperability and requires one to identify functionalities able to support interoperability of locally adaptable systems by mediating among work processes at the modeling level too. That is, the necessity to 'reconcile' different views of distributed teams on shared or borderline entities has to become part of the construction of the interoperating coordination mechanisms that support their cooperation. This concept is rooted in following theories about workflow organization.

5.3 Holonic systems

The Holonic Enterprise (HE) has emerged as a business paradigm from the need for flexible open reconfigurable models able to emulate the market dynamics in the networked economy, which necessitates that strategies and relationships evolve over time, changing with the dynamic business environment as stated in [13], [22] and [18].

The main idea of the HE model stems from the work of Koestler, [12]. In his attempt to create a model for self-organization in biological systems, Koestler has identified structural patterns that form nested hierarchies of self-replicating structures, named holarchies. Koestler proposed the term "holon" to describe the elements of these systems. This term is a combination of the Greek word holos, meaning "whole", with the suffix -on meaning "part", as in proton or neuron. This term reflects the tendencies of holons to act as autonomous entities, yet cooperating to form apparently self-organizing hierarchies of subsystems, such as the cell/tissue/organ/system hierarchy in biology.

Holons interaction Holons at several levels of resolution in the holarchy behave as autonomous wholes and yet as cooperative parts for achieving the goal of the holarchy. Within a holarchy, holons can belong to different clusters simultaneously, displaying rule-governed behavior. The rules define a system as a holon with an individuality of its own; they determine its invariant properties, its structural configuration and functional pattern. The duality autonomy-cooperation is a main contradictory force within a holarchy. From a software engineering perspective, a holon, as a unit of composition retaining characteristic attributes of the whole system (holarchy), can be viewed as a class.

MAS vs Holonic systems

A software representation of a holarchy appears natural as MAS, consisting of autonomous yet cooperative agents. From this perspective a MAS is regarded as a system of agents (software holons) which can cooperate to achieve a goal or objective. The MAS (software holarchy) defines the basic rules for cooperation of the agents (software holons) and thereby limits their autonomy.

In this context, autonomy is defined as the capability of an entity (i.e. agent or holon) to create and control the execution of its own plans and/or strategies, while cooperation is the process whereby a set of entities develop mutually acceptable plans and execute them. The common denominator between holonics and MAS as paradigms is obviously the focus on the dynamics of the interactions. However, in a MAS there is no pre-assigned condition that the interactions should be driven by cooperative forces, while in a holonic system this is a precondition for the existence of the holarchy per se (the glue that binds the holarchy together driving it towards the common goal).

Holarchy team-spirit This "team- spirit" characterizes a holarchy, in that all its component parts at all levels of resolution work together towards achieving the goal in an optimal manner. This "togetherness" drives the self-organizing power that configures all the sub-holons to optimize the interactions within the holarchy to reach the common goal with maximum efficiency.

Web-centric model vs Virtual Enterprises (VE)

Enterprises partially "cloned" as agents that interact over the Internet, can cluster as well into holarchies to form global virtual organizations. Two main enterprise-related paradigms have emerged supported by this technological development: the Web-centric enterprise and the virtual enterprise.

The Web-centric enterprise model At the core of the Web-centric enterprise model is the Internet-enabled software infrastructure acting as a worldwide open DSE. Such an integrated framework enables sharing of information, services and applications among suppliers, employees, partners and customers via: Deployment of automated, intelligent software services (e.g. Internet-enabled negotiations, financial transactions, advertising and bidding; order placement/ delivery, etc.). Complex interactions between such services (e.g. compliance policies; argumentation and persuasion via complex conversation protocols, etc.). The Web-centric model addresses product and process life-cycle management across the extended enterprise regarded as a global organization.

Virtual Enterprises A virtual organization or company is one whose members are geographically apart, usually working by electronic linking via computers while appearing to others to be a single, unified organization with a real physical location. Within a virtual organization, work cannot be completed without support of an information technology infrastructure in linking the parts. VE is a distinct organizational form, not just a property of any organization. Thus, Web-centric organizations that can use communications extensively, but not in a way critical in fulfilling the goal of the organization (e.g. a multinational corporation with dispersed parts being on the same satellite network whose use, however, is not critical for completing the production process) are not VE. The VE is an appropriate model for strategic partnerships.

5.4 Wfms and Virtual Enterprises

Since Virtual enterprises are temporary aggregation of autonomous and heterogeneous enterprises connected through a network. The main issue they have to deal with is related with dependencies complexity and coordination problems. In fact, the activities are defined by agents that involve the interaction with each others. To resolve this issue, wfms use technology as tuple centre model that allows to design and implement flexible agent-based wfms through an infrastructure like TuCSon. It uses tuple centre as wf engine that encapsulates rules as wf using ReSPeCt, a specification language, general-purpose interpreter for wf specifications. TuCSon guarantees portability, support different middleware and heterogeneous process facing coordination of dynamic VE aspects.

The importance of the middleware The coordination among different organization wf is possible using a middleware, that enables interaction between autonomous distributed components that hide differences providing a common interface for both application and components (as OS). In fact coordination is succesful if it's invisible through distributed systems that has 3 principal characteristics: composabliity, the best way to share info among indipendent activities,

ownership and alterity. To support these points an abstract architecture based on coordination infrastructure is built: main wf node contains the distributed wf general specification, the number of nodes distributed across wf participants and newtork of wf agents exchange among wf engines to maintain consistency.

5.4.1 Inter-Organizational Workflows

Inter-organizational workflow flexibility is defined by cooperative different activities/actors. Seeing that wfms appear too rigid and it is difficul to apply semantic interoperability because wf must adapt to business evolution/changes. The main related issues are: interoperability, components systems difficulty to work with different ones, portability, how much an app can be moved to a different distributed system and working, extensibility, difficulty to add comopnents and functionalities to an existing distributed system.

IOW coordination thorugh Multi-Agent Systems is defined by artificial participants that are responsible for automated tasks execution, intelligent interfaces with respect to the wf system, encapsulating the tasks and individual activities execution. Coordination artifacts are like target of the agents activities, computational objects or tools that agents use to support their work capturing structure and function of wf engines (ef activities coordination). They also support awareness, coordination and mutual knowledge managing dependencies and interactions occurring among agents.

5.5 Self-configuration

Frequently, workflow management systems are applied to domains having unpredictable workload. Manual configuration of workflow execution systems is difficult because: workload characteristics are not known a priori, misconfiguration leads to performance penalty. So, how to configure a workflow distributed engine? As stated in [11], autonomic principles can be applied to a distributed workflow engine to overcome this issue.

5.5.1 Tuple Space Application

It is possible to scale the system to run on a cluster of computers, as navigators and dispatchers threads can be physically located on different nodes. However, at most one thread (dispatcher or navigator) is running on a node at a given time. Thanks to the flexibility provided by tuple spaces, it becomes feasible to dynamically reconfigure the system, as the number of navigators and dispatchers can be increased or decreased without having to stop the whole system. To do so, the system offers a reconfiguration API that makes it possible to control which thread is running on each node of the cluster. Tuple spaces also offer a convenient mechanism for instrumenting the system in order to gather performance information that can be fed back into the self-tuning algorithm of the autonomic controller.

5.5.2 Autonomic capabilities

Autonomic computing refers to the self-managing characteristics of distributed computing resources, adapting to unpredictable changes while hiding intrinsic

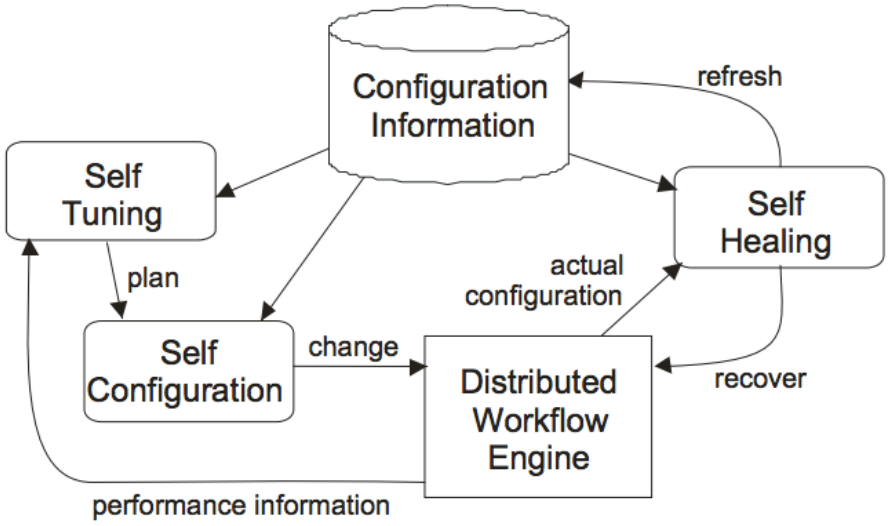

Figure 5.2: Autonomic Controller overview

complexity to operators and users.

Self-Tuning

The self-tuning component is responsible for determining whether the current system configuration is optimal. In case an imbalance is detected and a change of configuration is needed, the self-tuning component submits a reconfiguration plan to the self-configuration component. To do so, the self-tuning component acts upon three different strategies.

Information Strategy What performance indicators should be monitored to determine a new configuration? The growth of the task execution resquest space (waiting tasks space) size as well as the process execution request space (waiting processes space) size and the current configuration information.

Optimization Strategy What is the best configuration to balance the number of dispatcher and navigator threads? The configuration is adapted such that growth of either space is bounded.

The selection strategy How to map the reconfiguration decision onto the cluster.

Self-Configuration Actions

The self-configuration actions mechanism performs several actions: starting Threads, stopping Navigator Threads, stopping Dispatcher Threads and involves stopping the execution of tasks. To do this, it adopts two different methods: kill method that stops all tasks immediately and stop method that lets all

tasks finish but do not start new tasks. However, stopping a dispatcher may introduce reconfiguration overhead, this system uses 2 techniques to overcome this problem: random selection selects dispatcher threads arbitrarily and smart selection selects dispatcher that introduces the least reconfiguration overhead.

Self-Healing

Failures are detected by comparing the current configuration of the execution engine with the configuration information. The system handles dispatcher thread failures quering the state of the execution of the process to determine which tasks need to be re-executed, then requires tasks running on the failed dispatcher to be rescheduled. Moreover, it handles navigator thread failures enabling execution state vailable in the global process execution state space.

5.6 Workflow adaptation

Workflow adaptation involves 4 principal kinds of elements: infrastructure, resource, process and domain. Formal constructs (es. wf, procedures,..) have an important role to support coordination in organization. In fact, they help to identify the requirements of a computer system. Especially in the cooperative ensembles that act in business environment that are dynamic and uncertain. Formal constructs are strongly related to action situadness and are mainly divided into 2 elements: maps that are references to orientation purposes and scripts that are normative constructs based on precomputation of interdependencies with specific stipulation that instruct actor to take actions according to rules in protocol. Maps rules can be modified and adapted at reasonable cost whereas scripts rules are specified by actors during action. Together allow requirement adaptability obtained through combination of different dimensions that complement each other:

- Language. It models the formal constructs that shape the work processes. Adaptability: tailor and use the most suitable modelling language.
- Visibility of context of definition. Adaptability: availability of environments that provide different levels of visibility of the construct in relation to the need of actors flexibly.
- Modifications of formal constructs based on requirements of dynamic and uncertain environment. Adaptability: support incremental design and change formal constructs, and tools supporting the propagation of changes in the environment.

5.7 AgentWork

Conventional wfms don't provide sufficient flexibility to cope with failures. Control/data flow of a wf is not adeguate anymore, it has to be adapted automatically to reduce time-consuming and errors. Muller, Robert and Greiner, [14], defined a wfms prototype to overcome limitations of existing systems and comprehensively support automated wf adaptations called AgentWork. It predictevely adapts the yet unexecuted parts of running wfs automatically. It uses:

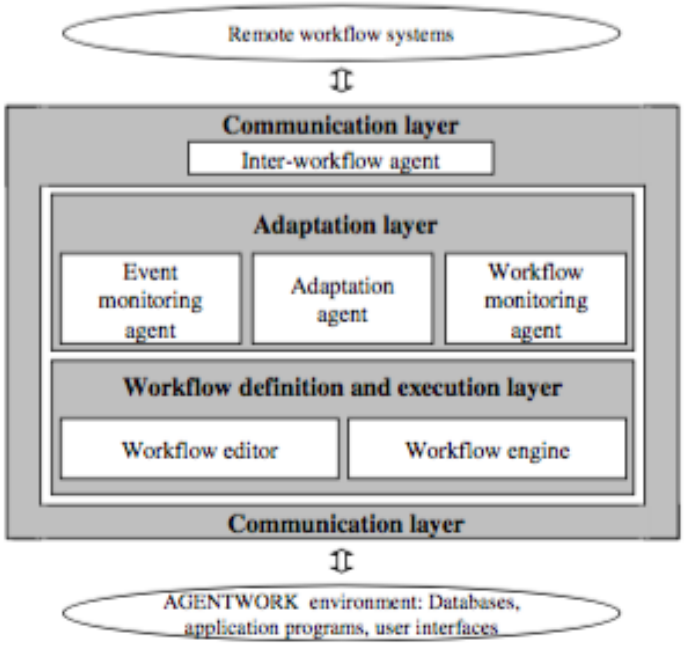

Figure 5.3: AgentWork overview

- ECA (Event/Condition/Action) rule mode to automatically detect logical failures and determine the necessary wf adaptations;
- WF estimation algorithm to determine which wf part is affected by a logical failure;
- a comprehensive set of operators for automatic wf adaptation, i.e. control flow operators, allow to add/delete wf activities;
- mechanisms to monitor adapted wfs by checking whether the used time estimated are met the adapted wf.

5.7.1 Architecture

It has a 3 layers architecture:

- Communication layer. It manages communication between AgentWork components and the environment (middleware CORBA + message format XML);
- Inter-workflow agent. It determines if a logical failure has implications on a wf and its other cooperated wfs;
- Adaptation layer. Its components are called agents, they have properties associated with agent-oriented modelling and programming (intelligence, autonomy, cooperation). There are 3 agents for logical failures handling:

  - Event monitoring agent. It decides which events constitute logical failures (with ECA);
  - Adaptation agent. It decides which adaptation is suitable and defines wf estimation;
  - WEf monitoring agent. It checks if assumptions of adapted agent are met.

- Wf definition and execution layer. It provides components for wfs definition and execution.

5.7.2 Workflow definition model

In AgentWork the workflow definition model is graph-oriented. Activities are represented by activity nodes associated to activity definition (activity details specification). The control flow is represented by edges and control nodes for conditional branching (OR-SPLIT/OR-JOIN), parallel execution (AND-SPLIT/AND-JOIN) and loops (LOOP-START/LOOP-END, symmetrical). Edges are synchronized, synchronizing nodes belonging to parallel control flow paths of a block.

5.7.3 Workflow adaptation and monitoring

The procedure that allow AgentWork to adapt and monitor workflow is the following:

1. Decide which adaptation strategy is suitable;
2. Provide mechanisms for the estimation of a wf's future behaviour;
3. Control actions transalted into structural wf adaptations taking into account the current execution state of running wfs;
4. Wf monitoring to control whether the adaped wf execution matches the temporal estimates.

5.8 Automated Workflows

Adaption and pertinence to the dynamic and diverse situations is a requisite for adoption of automated workflow guidance. Workflow problems:

1. The exorbitant cost of modeling diverse workflows results in the absence of existing workflow models and subsequently automated guidance for these types, yet these unique cases are often the ones where guidance is most desirable;
2. Rigid, pre-defined workflow models are limited in their adaptability, thus the workflows become situationally irrelevant and are thus ignored;
3. Entwining the complex modeling of situational property influences (as risk or urgency) on workflows within the workflows themselves incorporates an implicit modeling that unduly increases their complexity and makes correct maintenance difficult.

The cognitive effort required to create and maintain large process models syntactically can lower the attention towards the incorporated semantic problem-oriented content. This work focuses on applying context-awareness utilizing semantic processing and situational method engineering for automatically adapting workflows in a process-aware information system. Dynamic on-the-fly workflow generation and adaptation using contextual knowledge for a large set of diverse workflow variants is thus supported, enabling pertinent workflow guidance for workers in such environments.

5.8.1 Solution Approach

In [10], Grambow, Gregor and Oberhauser define a comprehensive automated process support to address the challenges mentioned before called COSEEK (Context-aware Software Engineering Environment Event-driven frameworK).

Software Engineering Environment components

Since workflows belonging to SE processes they deal with SE issues that are not modeled in those processes:

- Event Extraction consists of SE Tool sensor events (e.g., creation of a certain source code file) that are acquired and then stored in an XML Tuple Space, where it may optionally be annotated with relevant contextual information (e.g., link to a requirement for traceability).

- Event Processing detects higher-level events. This may result in workflow adjustment (e.g., according to the type of source code file, an activity Implement Solution or Implement Test may be chosen), and the software engineer is informed of a change in tasks via process management in their IDE (Integrated Development Environment). The Context Module includes an ontology and reasoner.

Process management requirement

Process Management requires an adaptable process-aware information system due to the dynamic nature of the problem the current approach seeks to address. It allows authorized agents to dynamically adapt and evolve the structure of process models during runtime. Such dynamic process changes do not lead to an unstable system behavior, i.e., none of the guarantees achieved by formal checks at build-time are violated due to the dynamic change at runtime.

Correctness is ensured in two stages. First, structural and behavioral soundness of the modified process model is guaranteed independent from whether or not the change is applied at the process instance level. Second, when performing structural schema changes at the process instance level, this must not lead to inconsistent or erroneous process states afterwards.

Application of Situational Method Engineering

Situational method engineering adapts generic methods to the actual situation of a project. This is done based on different influence factors called process

properties which capture the impact of the current situation, and product properties which realize the impact of the product currently being processed (for this context the type of component, e.g., a GUI or database component).

To strike a balance between rigidly pre-modeled workflows and no process guidance, the idea is to have a basic workflow for each case that is then dynamically extended with activities matching the current situation. The construction of the workflows utilizes a case base as well as a method repository. The case base contains a workflow skeleton for each different case. This workflow only contains the absolutely fundamental activities that are always executed for that case. The method repository contains all further activities whose execution is possible according to the case. To be able to choose the appropriate activities for the current artifact and situation, the activities are connected to properties that realize product and process properties of situational method engineering.

SE issue mapping Each SE issue, such as refactoring or bug fixing, is mapped to exactly one case relating to exactly one workflow skeleton. To realize a preselection of activities (e.g., Create Branch or Code Review) which semantically match an issue, the issue is connected to the activity via an n-m relation. The activities are in turn connected to properties specifying the dependencies among them. The selection of an activity can depend on various process as well as product properties. To model the characteristic of an issue leading to the selection of concrete activities, the issue is connected to various properties. The properties have a computed value indicating the degree in which they apply to the current situation. An example for a property would be 'risk' with a value of 'very high', marking that issue as a very high-risk issue. Utilizing the connection of activity and property, selection rules for activities based on the values of the properties can be specified.

Activity selection and sequencing

To be able to dynamically build up the workflow for an SE issue, after completing the computation of the property values activities have to be selected and placed in the correct order. This is done utilizing the connection between properties and activities.

The selection of activities results in an unordered list of activities that have to be correctly sequenced and inserted into the workflow skeleton. To guarantee this, CoSEEEK uses a set of simple semantic constraints. These constraints do not only enforce which activities are permitted in a particular workflow, but also determine their correct ordering. A predefined sequencing of the activities is required since the workflow is built up at the beginning of its execution. That implies that each of the activities must have a binary relation to all other activities so that every possible set of activities can be sequenced. For the time being, the approach presented requires a proper specification of these constraints and only deals with linear workflows.

Concrete Procedure

The concrete procedure for the handling of a SE issue in the presented approach is as follows:

1. As entry point for the workflow there is an event in the framework indicating that an SE issue is assigned to a user. This event can come from various sources. Examples include the assignment of an SE issue to a person in a bug tracker system or the manual triggering by a user via the GUI.

2. The next step is the determination of a case for that issue like 'Bug fixing' or 'Refactoring'. Depending on the origin of the event, th is can be done implicitly or explicitly by the user.

3. When the case is specified, the workflow starts for the user using the workflow skeleton assigned to that case, as does the contextual information-gathering phase for the properties of the case.

4. After having determined the properties for the case, the additional activities matching the current situation and product are selected.

5. Thie set of activities is then checked for integrity and correctly sequenced utilizing semantic constraints.

6. Subsequently, the activities are integrated into the running process instance.

If one or more of the properties change during the execution of the workflow, the prospective activities are deleted (if still possible) and a new sequence of activities is computed for the rest of the workflow.

5.8.2 Solution considerations

The synergistic CoSEEEK approach automatically adapts workflows in a process-aware information system by combining semantic-based SEE context knowledge with situational method engineering and automated process instance adaptations. SE issue processing is decomposed into various activities influenced by different process and product properties dependent on the actual situation, the project context knowledge, and the product that is the subject of the current SE issue. Based on these properties, an issue workflow is constructed automatically, dynamically, and uniquely for every SE issue. By combining a case base with a method repository, all activities that are requisite for an issue are automatically included, avoiding the necessity of building the current workflow from scratch.

The broader application of this approach would benefit domains similar to SE that exhibit dynamics and high workflow diversity with adaptable workflows for uncommon workflows, providing useable context-relevant guidance while reducing workflow modeling effort and maintenance by modeling influences outside of the workflows themselves. Future work will consider issue learning for automated tailoring of process templates and to reduce external user information needs, continuous adaptation of product properties, automated case-learning, and process analysis of executed workflow instances.

Chapter 6

Conclusion

Business and private life are becoming increasingly agile and chaotic and asked for tools development or new mechanisms for dealing with "procedural knowledge". In this paper I tried to delineate techniques and methods used and developed towards this trend. What I tried to point out is that studying from previous theories and concepts is possible to find ways to solve present issues. What's more, reading papers I had the impression that the research is approaching a new step in dealing with new challenges. In particular, I noticed an evolution in the way of thinking that brought the research to systems autonomy need. In last decade CSCW issues affected principally this theme, also in question where it could appear insignificant. But how to make systems autonomous? Simply, observing systems that are naturally autonomous: natural systems. Basic structures that are hidden below natural systems are being described and approaches that implement those structures are being illustrated. Otherwise, not only world developments don't stop but evolve spasmodically as well as CSCW evolution that must follow them.

Recently, new applications for CSCW and specifically wfms are emerged. These novelties include several fields as medicine, cookery, computer science and social science. In computer science, while the potential benefits of cloud computing are clear, there are still significant technical hurdles associated with obtaining the best execution efficiency whilst trading off cost ([9] and [1]). Moreover, thanks to the beginning of the new social era, social workflows were born and proposed "as an executable process representation, serving private people and groups of people to fulfill their objectives by providing means to store, create, and link personal activities and data objects according to procedural rules." [8] Since there was no possibility to link services to form a flow of activities involving different people such as friends or professionals, Gorg, Bergmann, Gessinger and Muller developed a framework called CAKE (Collaborative Agile Knowledge Engine). CAKE "is a generic software system for integrated process and knowledge management, which can be configured to different application needs" ([3]).

The constant technology evolution required a constant evolution of cooperative systems and elements included in their structure. Systems progressively look and are organised as natural systems: they autoregolate their internal state and interaction as well as interaction with others. It's clear we're going towards the autonomy era.

Bibliography

[1] V. Arabnejad and K. Bubendorfer. Cost effective and deadline constrained scientific workflow scheduling for commercial clouds. In *Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on*, pages 106–113. IEEE, 2015.

[2] J. E. Bardram. Plans as situated action: an activity theory approach to workflow systems. In *Proceedings of the Fifth European Conference on Computer Supported Cooperative Work*, pages 17–32. Springer, 1997.

[3] R. Bergmann, S. Gessinger, S. Gorg, and G. Muller. The collaborative agile knowledge engine cake. In *Proceedings of the 18th International Conference on Supporting Group Work*, pages 281–284. ACM, 2014.

[4] M. Divitini and C. Simone. Supporting different dimensions of adaptability in workflow modeling. *Computer Supported Cooperative Work (CSCW)*, 9(3-4):365–397, 2000.

[5] M. Divitini, C. Simone, and K. Schmidt. Abaco: Coordination mechanisms in a multi-agent perspective. In *COOP'96. Second International Conference on the Design of Cooperative Systems, Antibes-Juan-les-Pins, France, 12-14 June*, pages 103–122, 1996.

[6] J. H. et al. (eds.). Aether: An awareness engine for cscw. In *Fifth European Conference on Computer Supported Cooperative Work*, pages 221–236, Netherlands, 1997. Kluwer Academic Publishers.

[7] E. Fimiani. Workflow management. Esame Sistemi Distribuiti.

[8] S. Gorg and R. Bergmann. Social workflows—vision and potential study. *Information Systems*, 50:1–19, 2015.

[9] S. Görg, R. Bergmann, S. Gessinger, and M. Minor. Real-time collaboration and experience reuse for cloud-based workflow management systems. In *2013 IEEE 15th Conference on Business Informatics*, pages 391–398. IEEE, 2013.

[10] G. Grambow, R. Oberhauser, and M. Reichert. Semantic workflow adaption in support of workflow diversity. In *4th Int'l Conf. on Advances in Semantic Processing*. SEMAPRO'10, 2010.

[11] T. Heinis, C. Pautasso, and G. Alonso. Design and evaluation of an autonomic workflow engine. In *Second International Conference on Autonomic Computing (ICAC'05)*, pages 27–38. IEEE, 2005.

[12] A. Koestler. The ghost in the machine. 1967. *London: Hutchinson*, 1967.

[13] P. McHugh, G. Merli, and W. A. Wheeler. *Beyond business process reengineering: towards the holonic enterprise*. John Wiley & Son Ltd, 1995.

[14] R. Muller, U. Greiner, and E. Rahm. Agentwork: a workflow system supporting rule-based workflow adaptation. *Data & Knowledge Engineering*, 51(2):223–256, 2004.

[15] G. J. Nutt. The evolution towards flexible workflow systems. *Distributed Systems Engineering*, 3(4):276, 1996.

[16] A. Omicini. Corso "sistemi autonomi". *https://apice.unibo.it/xwiki/bin/view/Courses/Sa1516*, Campus di Cesena - Università di Bologna (Unibo), 2015.

[17] A. Riccia, A. Omicinia, M. Virolia, L. Gardellia, and E. Olivaa. Cognitive stigmergy: A framework based on agents and artifacts. In *Environments for Multi-Agent Systems III*, pages 124–140. Springer, 2006.

[18] M. Schillo and K. Fischer. Holonic multiagent systems. *Manufacturing Systems*, 8(13):538–550, 2002.

[19] K. Schmidt and C. Simone. Mind the gap. *Towards a unified view of CSCW. COOP*, pages 205–221, 2000.

[20] L. F. Steve Benford. A spatial model of interaction in large virtual environments. In *Proceedings of the Third European Conference on Computer-Supported Cooperative Work*, pages 109–124, Milan, 1993. ECSCW '93.

[21] L. Suchman. Do categories have politics? *Computer Supported Cooperative Work (CSCW)*, 2(3):177–190, 1993.

[22] M. Ulieru, R. W. Brennan, and S. S. Walker. The holonic enterprise: a model for internet-enabled global manufacturing supply chain and workflow management. *Integrated Manufacturing Systems*, 13(8):538–550, 2002.