Chapter 3
Towards Executable Specifications for Case Management Processes

Irina Rychkova, Bénédicte Le Grand and Carine Souveyet

Abstract Explicit process specifications play an important role in process-aware information systems (PAIS). Whereas methodologies for modeling structured, activity-oriented processes are well established, modeling formalisms for unstructured processes such as case management processes (CMP) are lagging. In this chapter, we define a state-oriented formalism that allows for executable specifications of CMP and paves the road for predictive analysis and recommendations support intended to case managers. This formalism is grounded on statecharts developed by D. Harel in 1987. We adopt the main concepts defined by statecharts and demonstrate how they can be used to specify a case management process. We also propose adaptations and potential extensions of the statecharts formalism that could address CMP specifics and complexity.

Keywords Case management · Simulation-based testing · Automated process analysis · Recommendations · State machines · Statecharts

3.1 Introduction

A Process-Aware Information System (PAIS) is a software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of process models [16]. Workflow management systems and BPM systems are classic examples of PAIS.

I. Rychkova (✉) · B. Le Grand · C. Souveyet
Université Paris 1 Panthéon-Sorbonne, 12, Place du Panthéon, 75005 Paris, France
e-mail: irina.rychkova@univ-paris1.fr
URL: http://www.univ-paris1.fr/centres-de-recherche/cri/

B. Le Grand
e-mail: benedicte.le-grand@univ-paris1.fr

C. Souveyet
e-mail: carine.souveyet@univ-paris1.fr

G. Grambow et al. (eds.), *Advances in Intelligent Process-Aware Information Systems*, Intelligent Systems Reference Library 123,
DOI 10.1007/978-3-319-52181-7_3

Started by F. Taylor and H. Ford, a pursuit of process optimization and automation resulted in the creation of workflow concepts, where a process is specified with a (predefined) flow of tasks [55]. Workflows provide a powerful formalism for the design, simulation, analysis as well as management and execution of s*tructured, activity-oriented processes*.

Today, practitioners express the increasing need for information systems supporting *unstructured, data-oriented processes* such as *case management processes (CMP)*. The Object Management Group (OMG) defines case management as "a coordinative and goal-oriented discipline, to handle cases from opening to closure, interactively between persons involved with the subject of the case and a case manager or case team" [34]. Davenport [12] defines a case management process as a process that is not predefined or repeatable, but instead, depends on its evolving circumstances and on decisions regarding a particular situation, i.e., a case. Claim processing, residence permit issuing, crisis management, and organization of events are examples of CMP.

PAIS supporting case management are gaining momentum nowadays. Among successful solutions the IBM Advanced Case Manager,[1] ISIS Papyrus,[2] Computas[3] or IBM Intelligent Operations Center[4] can be cited. Many solutions supporting case management are now being developed and reported by the community of practitioners promoting Adaptive Case Management (ACM) [49].

Explicit process specifications play an important role in PAIS: they allow for better communication between stakeholders, enable process analysis and support redesign efforts [2]. Methodologies, specification languages and environments for workflow modeling and analysis are widely presented in the literature and recognised by practitioners. In contrast, current CMP supporting solutions are mostly focused on process configuration and execution. Very little support for CMP modeling and analysis is provided.

In this chapter, we define a state-oriented formalism for the incremental and interactive modeling and simulation of CMP. Our formalism is grounded on statecharts developed by D. Harel in 1987 [19]. In particular, we explain (a) why statecharts is a suitable formalism for CMP, (b) how statecharts can be adOpted and adApted for specifying CMP; we also show (c) how executable statecharts specifications can be used for CMP simulation and (d) how they can enable predictive analysis and recommendation support for a case manager.

Statecharts were originally created as a visual, fully executable formalism for the specification, design and analysis of complex discrete-event systems. Case management processes share a number of characteristics with complex discrete-event systems [19, 20, 23]: they continuously interact with their environment, respond to

[1] http://www-03.ibm.com/software/products/en/category/advanced-case-management.

[2] http://www.isis-papyrus.com/.

[3] http://www.computas.com/.

[4] http://www-03.ibm.com/software/products/en/intelligent-operations-center.

unexpected events (interrupts) and have many possible operation scenarios. In particular, a CMP can be compared to a reactive system, for which the main challenge is to identify the appropriate activity or group of activities to perform in reaction to a given internal or external stimulus in a given situation (context). However, contrary to conventional reactive systems, CMP has a *goal* that can be reached by several alternative scenarios. Moreover, decisions about these scenarios in CMP are typically made by a human actor (the case manager). Therefore, a CMP supporting system can seldom automatically execute the activities but it can enable or recommend them for execution.

The statecharts formalism combines an intuitive and concise visual notation with precise semantics [21, 31]. Rhapsody [20] (now IBM Rational Rhapsody[5]) and the open source YAKINDU Statechart Tools (SCT)[6] are examples of statecharts modeling environments, where visual statecharts specifications can be created and executed.

Following the points stated above, *we adopt the main concepts of statecharts*, such as states and state hierarchies, transitions, triggering events, concurrency and broadcast communication for CMP specification. In order to address CMP specific features, *we extend the statecharts formalism* with the notions of goal and path; we also revisit the semantics behind triggering events and introduce the concept of event duration.

The advantages of statecharts specifications can be perceived both during the design of CMP and during their execution. As we will explain in this chapter:

- Statecharts specifications allow for incremental CMP design;
- Executable statecharts specification can be used for the simulation-based testing of CMP scenarios;
- Executable statechart specifications pave the road for automated recommendations for CMP.

We apply the proposed formalism to specify an example of CMP: a crisis (flood) management process defined for Hauts-de-Seine department of France.

The remainder of this chapter is organized as follows. In Sect. 3.2, we introduce our example and provide the terminology that will be used in this chapter. This terminology spans across two domains: complex systems and case management. In Sect. 3.3, we present and discuss the related work in CMP management and modeling. In Sect. 3.4, we introduce the statecharts formalism and draw the parallels between complex discrete-event systems and CMP. In Sect. 3.5, we demonstrate how the statecharts formalism can be adopted and extended in order to provide fully executable specifications of CMP. In Sect. 3.6, we discuss the prospective added value of executable CMP specifications, trace a roadmap for future research and draw our conclusions.

[5] http://www-03.ibm.com/software/products/en/ratirhapfami.

[6] http://statecharts.org/index.html.

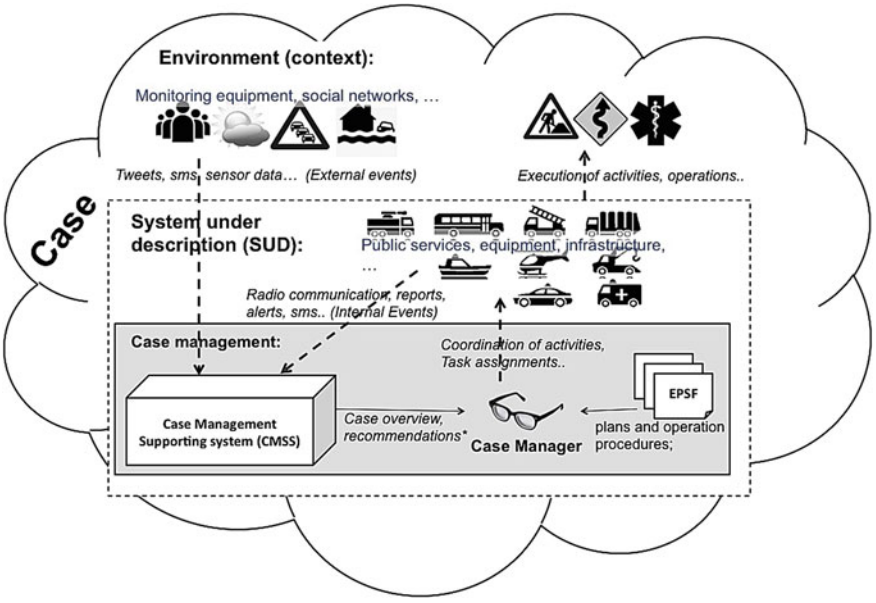

Fig. 3.1 The scope the flood management process

3.2 Case Management Process Example and Terminology

In this section we provide an example of CMP—a crisis management process designed to handle floods (we will call it flood management process) in a French department Hauts-de-Seine. We also briefly introduce the terminology used in this chapter and illustrate it on our example. Figure 3.1 shows the scope of the flood management process.

3.2.1 Crisis Management in Cases of Flood

A *flood* is an overflow of water that submerges a land. It happens, for example, because of an increase in the flow of a river provoked by significant rainfalls. The risk of a "major flood" is the main natural risk in the Ile-de-France region, particularly during the winter period from November to March. Cities like Paris[7] are confronted to this risk: if a flood occurs, important damages can be expected, affecting thousands of people. Floods are considered harmful when the water level of the Seine river

[7] See http://cartorisque.prim.net/dpt/75/75_ip.html.

exceeds 5.50 m according to the scale on the Austerlitz bridge in Paris. In the Hauts-de-Seine department, the risk of flood is considered as particularly important since 1910.[8]

The goal of the flood management process is to maintain the proper operation of city infrastructure (water supply, electricity, telecommunication, road networks, public transport and so on) and to protect people and facilities from flood consequences. This process is a typical example of CMP:

- it demands interaction between multiple actors (government, public services, volunteers, etc.).
- it is driven by the dynamic context of the case (i.e., flood development, current status of vulnerable areas and of rescuing operations) rather than by a predefined sequence of activities.

Flood Emergency begins when the water level rises above 5.5 m at the Austerlitz Bridge and is supposed to keep rising (according to weather forecasts). At this stage, the centers for crisis management are set up and the Emergency Plan Specialized on Floods (EPSF) is triggered. The city services (rescue, fire fighters, police, etc.) therefore carry out specific activities accordingly.

The regional authorities monitor the crisis situation and coordinate the operation procedures in the following major areas: *evacuation of population and facilities, temporary accommodation, public transport, road traffic, water supply, electricity supply and telecommunications.*

According to the flood severity, the EPSF identifies different phases of flood emergency for each of these seven areas and specifies the procedures to control the situation and to protect the population and facilities.

For example, when the water level exceeds 6.25 m, the drinking water supply is reduced for the towns of Saint-Cloud, Garches, Vaucresson, Marnes la Coquette and Ville d'Avray. When the water level reaches 6.7 m, the drinking water supply for these towns is completely disrupted. Therefore, the provisioning and distribution of bottled drinking water should start as soon as the water level at Austerlitz Bridge reaches 6.25 m. In case of limited supply, prioritized water provisioning has to be organized.

Along those lines, depending on the water level, various procedures are launched: a partial or complete interruption of public transport (SNCF Paris Rive Gauche, RER C, RATP), deviation and blocking of main highways (A86, A14, N14, etc.), evacuation of people, health care and childcare facilities.

Resources available for crisis management also need to be constantly monitored. In case of deficiencies in equipment, manpower or other problems that can compromise the crisis handling in one or several areas, specific measures such as mobilization of volunteers or federal alert raising can be taken. We model the resources as a specific area of the EPSF.

---

[8] Source: Préfecture des Hauts-de-Seine: Plan de secours spécialisé sur les inondations Hauts-de-Seine, SIDPC 21/11/2005, (2005), Available at: http://www.ville-neuillysurseine.fr/files/neuilly/mairie/services_techniques/plan-secours-inondation.pdf.

3.2.2 Terminology Used in This Chapter

A *case* is a situation (e.g., a flood crisis), which requires resolution. It is described by a set of elements that are relevant to or involved in a CMP. Within the case, we define the system boundary and distinguish between so-called system elements and context elements (that belong to the environment):

Case = System under description + Environment

The *System Under Description (SUD)* is described by the set of elements that can be controlled during the case management: public services, equipment, infrastructure, administration etc. It also includes a Case Management Supporting System (CMSS) and a case manager.

The SUD reacts to various stimuli (events) produced by the environment (e.g., change in temperature, water level, incidents) and performs activities in order to maintain the functioning of city infrastructure and to protect people and facilities from flood consequences.

The SUD produces internal events such as messages, reports and alerts sent by the agents via radio or mobile network. They can indicate the success or failure of a mission, resource deficiencies, emergency situations and so on.

The *environment* is described by the set of elements that interact with the SUD. It cannot be controlled but only monitored using specific equipment (e.g., meteo stations for monitoring weather, embedded sensors for measuring water level, video cameras for measuring traffic, social networks for collecting information about areas affected by the flood). The environment's behavior is unpredictable and brings uncertainty in the CMP.

The environment produces external events such as accidents, traffic jams, electric outages, malfunctioning of telecommunication.

The *Case Management Process (CMP)* describes the behavior of the SUD and defines what it has to do in order to achieve some objectives, i.e., to ensure safety and security for people and goods during the flood, until the emergency is over.

The *case management* element in Fig. 3.1 depicts a subsystem of the SUD which is responsible for the coordination of SUDs activities. It includes the case manager and the case management supporting system (CMSS):

The *Case Management Supporting System (CMSS)* is a PAIS for case management. The *case manager* is a human actor who uses the CMSS in order to monitor the case, to take decisions regarding the case handling scenario and to coordinate the activities of the SUD.

3.3 Related Work

In this section, we discuss Adaptive Case Management—for now, the most prominent paradigm for CMP support. We also review the existing modeling paradigms and formalisms for process specification and their capacity to model CMP.

3.3.1 Adaptive Case Management

The concept of Adaptive Case Management (ACM) has been defined as an "information technology that exposes structured and unstructured business information (business data and content) and allows structured (business) and unstructured (social) organizations to execute work (routine and emergent processes) in a secure but transparent manner".[9]

One of the major challenges identified by the ACM community, is the attempt to deal with CMP in the industry the same way as with regular business process—i.e., representing a case management by a workflow and focusing on the (predefined) *sequence of tasks*. This view implies that the data emerges and evolves within a process according to a predefined control flow similarly to a product evolving on a conveyor belt.

According to ACM [52], CMP must be organized around *a collection of data artifacts* about the case; the tasks and their ordering shall be adapted at run time, according to the evolution of the case circumstances and case-related data [41].

The body of knowledge on ACM has been extensively developed by practitioners; the best solutions are regularly reported in the book series on WfMC Global Awards for Excellence in Case Management [53, 54]. However, methodologies and formalisms for CMP modeling are rarely discussed.

3.3.2 Modeling Paradigms for CMP Specification

The important role of modeling in PAIS is discussed in [2]. The following general process modeling paradigms are identified in the literature [10, 11, 15]: activity-oriented, product (or state)-oriented and decision (or goal)-oriented.

The choice of a modeling paradigm depends on the conceptual properties of the process (e.g., flexibility vs. control).

According to the literature, case management processes (CMP) have the following conceptual properties:

1. CMP are unstructured, with non-repeatable execution scenarios [9, 52];
2. CMP are data-centered and are organized around a collection of data artifacts about the case [6, 52];
3. CMP are reactive and event-driven: activities should be carried out in reaction to a given internal or external event;
4. CMP must be considered within their context and the boundary between the system and its environment and the scope of the process must be clearly specified [6, 27, 48];
5. CMP are goal-oriented and flexible: goals are set and can be modified, added or removed during the execution [48];

[9] http://www.xpdl.org/nugen/p/adaptive-case-management/public.htm.

6. CMP are knowledge-intensive: decisions about the process scenario are made by a human actor—a knowledge worker—and are based on her knowledge, experience and intuition [25, 41];
7. CMP are unpredictable—they have to deal with events and handle the situations that were not planned or even imagined before [52].

In this section, we discuss the capacity of activity, product (state) and goal-oriented paradigms to express these conceptual properties of CMP.

Within the *activity-oriented paradigm*, the process is specified as an ordered set of activities that the system has to carry out. Examples of activity-oriented formalisms include BPMN [35], YAWL [3], activity diagrams in UML [46] and other languages based on workflow concepts.

Activity-oriented process modeling implies that data emerges and evolves within a process according to a predefined control flow. Events are supposed to occur (or be processed) at specific moments of the execution predefined by the model. This paradigm suits predictable and highly repeatable processes. CMP are unpredictable processes [52]: events and process inputs can occur at any time during execution; the order of activities cannot be predefined and depends on the current situation. Such behavior can therefore not be captured by the workflow formalism.

In order to increase process flexibility and to better address unstructured and knowledge-intensive processes like CMP, activity-oriented formalisms are extended with declarative parts, such as constraints [5], business rules [7] or configurable elements [45]. These formalisms can handle process variability within a potentially large number of configurations or scenarios. However, either such scenarios must be well identified upfront or the set of business rules (or configuration elements) must be regularly maintained by an expert. This can be seen as a limitation for CMP.

Techniques and frameworks for the analysis of activity-oriented process models are widely presented in the literature [57]. To provide automated process analysis, activity-oriented modeling languages are often annotated with or translated to some formal specification languages. The Declare framework [37] is a constraint-based system that uses a declarative language grounded on temporal logics. In [1], the state-oriented formalism of Petri Nets is used for workflow specification and analysis. In [14], the Petri Nets semantics for BPMN is presented. In [28], a business process model is mapped into a nondeterministic state machine for further analysis.

According to the *product-oriented (or state-oriented) paradigm*, a process is seen as a product life cycle (a set of product states and transitions between these states). Examples of product-oriented modeling formalisms include statemachines in UML [20], generic state-transition systems or state machines, such as FSM [38] or Petri Nets [32], and statecharts by D. Harel [19] created for the specification and analysis of complex discrete-event systems.

Within this paradigm, carried out activities depend on the current state of the product and the process scenario is adapted at run time, according to the evolution of the product. This paradigm suits well reactive systems specification [23] since the system's response to an event shall be defined not only by the type of this event but also by the current situation of the system i.e., its state.

Several research groups are reporting on approaches to design and specification of unstructured, knowledge-intensive processes (including CMP) based on the product-oriented paradigm.

In [9], process instances are represented as moving through a state space, and the process model is represented as a set of formal rules describing valid trajectories. Compared to our proposal based on statecharts, this approach is grounded on the theory of automated control systems. In [24], a group of researchers from IBM incorporates process- and data-centered perspectives; their approach is based on the concept of business artifacts. The Case Management Model and Notation is presented in [36]. This specification "is intended to capture the common elements that Case management products use, while also taking into account current research contributions on Case management." In [42], the Product-Based Workflow Design is presented. This approach explores the interaction between a product data model that reflects the product design and the process to manufacture this product represented by a workflow. The authors of [6] present case handling as a paradigm for supporting knowledge-intensive business processes. They recognise the lack of flexibility of workflow management systems and acknowledge the important role played by the "product"—the case—in the case handling. Their view on the case, however, remains activity-oriented: the proposed case definition explicitly includes the list of activities and their precedence relation assuming that they are known in advance.

Formalisms based on state machines are suitable for automated analysis including simulation, formal validation and model checking. Algorithms from graph theory can be used in order to analyse states reachability, "dead" states, path search and optimisation (where the path represents a process execution scenario).

In [24], the operational semantics of Guard-Stage-Milestone is presented. This semantics explains the interactions between business artifacts which are formalized following declarative principles. In our earlier work [47], we define formal semantics for CMP using the Alloy specification language. The Alloy Analyzer tool allows us to simulate and validate a CMP model; it also provides visual diagrams. Compared to statecharts, however, Alloy model is difficult to construct.

The product-oriented paradigm seems to be a good choice for specifying CMP. However, it does not support decision making since it does not define a notion of objective or goal.

The *decision or goal-oriented paradigm* extends the product-oriented view on the process: the successive transformations of the product are looked upon as consequences of decisions leading to some goal [33].

Goal-oriented modeling formalisms support decision making by specifying goal hierarchies and tracing each decision within these hierarchies. The examples include i* [58], KAOS [30], MAP [43].

Goal-oriented formalisms extended with the notion of context are presented in [40, 44, 51]. These formalisms link a decision (expressed as a goal) to the situation in which this decision is taken (product state): for each state, a set of achievable and non-achievable goals can be identified and vice versa, each goal can be expressed in terms of states that the product has to reach. These formalisms can also connect the goals and the activities that must/can be carried out in order to achieve these goals.

The Generic Process Model (GPM) [51] is an example of context-driven goal-oriented formalism. It captures the process context and allows for reasoning about process goals. It is also suitable for automated process analysis.

Context-driven goal-oriented process models support automated recommendations and user guidance, providing that for each goal all the situations (states) in which this goal is achievable are known. Due to unpredictable sequences of events and non-repeatable execution scenarios in CMP, however, it will be hard if at all possible to model relations between various process situations, goals and activities that must/can be executed in order to achieve these goals. Such relations can be, though, discovered using process mining techniques (this is an interesting subject that lies behind the scope of this work).

Our analysis of existing modeling paradigms and their corresponding formalisms shows that the activity-oriented paradigm can hardly provide the flexibility required by CMP as expressed by their conceptual properties 1–3 and 5–7 listed above. Configurable specifications and business rules can be used to overcome the rigidity of traditional workflow-based formalisms, addressing properties 5 and 1–3 respectively. Nevertheless, they support the variability of process scenarios only within some boundaries defined by a number of business rules or configurable elements. Thus, they fail to address properties 6 and 7 of CMP.

The goal-oriented paradigm offers flexibility and supports knowledge workers. However, goal-modeling formalisms are typically suitable for an early phase of system modeling (abstract system design); formal analysis, simulation and testing are not their priorities. Addressing properties 1 and 7 of CMP would lead to an extremely complex model.

The product-oriented (or state-oriented) paradigm addresses all conceptual properties of CMP except the 5th one—goal orientation—as this paradigm does not define the notion of goal. On the other hand, compared to goal-oriented formalisms, state-oriented modeling formalisms typically focus on concrete system design followed by validation and testing. They are supported by a plethora of techniques and tools for model simulation and formal analysis. Therefore, for modeling CMP, we adhere to the product-oriented paradigm.

3.4 Finite State Machines, Hierarchical State Machines and Statecharts

As explained above, we have chosen the product-oriented paradigm for modeling CMP. According to this paradigm, a state transition system (or state machine) represents our knowledge about the case and its evolution.

The choice of a concrete modeling formalism within the selected paradigm is related to the purpose of modeling (e.g., communication support, high-level design, simulation, formal validation and verification [18], diagnostics and improvement, recommendation and optimisation of process behaviour [8, 13]).

In this section we discuss a selection of existing formalisms based on state machines and focus on statecharts for CMP specification.

3.4.1 CMP Versus Complex Discrete-Event Systems

A CMP shares the following characteristics of complex reactive systems behavior defined in [19, 22]:

1. It continuously interacts with its environment. Its inputs and outputs are often asynchronous: they may occur or evolve unpredictably, at any time;
2. It must be able to respond to high-priority events (interrupts);
3. It has to operate and to react to inputs with respect to strict time regulations;
4. It has many possible operation scenarios, depending on its current mode of operation, current values of data as well as its past behavior;
5. It is very often based on interacting processes that operate in parallel.

As in a reactive system, the main challenge for the case manager is to identify the appropriate activities to perform in reaction to a given internal or external stimulus in a given situation (context).

State machines are a popular choice for specifying the behavior of reactive software systems. We will therefore consider them further.

3.4.2 Finite State Machines

A finite state machine (FSM) [38] specifies a machine that can be at one state at a time and can perform a state transition as a result of a triggering event (or a group of events guarded by a condition). It is defined by a (finite) set of states and a set of triggering events for each transition. To trigger a state transition, the execution of some activities and/or the observation of some contextual events can be required.

Traditional FSMs and their corresponding state-transition diagrams are very efficient for tackling small problems. However, the complexity of a FSM model tends to grow much faster than the complexity of the problem it represents. This makes the simulation or automated reasoning about the model extremely difficult. This phenomenon is called *the state explosion problem* [56].

3.4.3 Hierarchical State Machines and Statecharts

The state explosion problem can be overcome by the introduction of multiple hierarchical levels for states and transitions. Indeed, this hierarchy gives a possibility to reuse some common behaviors across many states and, thus, to reduce the model

complexity. This idea is explored in the formalism of statecharts, invented by David Harel in the 1980s [19].

The statecharts formalism specifies a hierarchical state machine (HSM); it extends classical FSM by providing:

(i) depth—the possibility to model states at multiple hierarchical levels, with the notion of abstraction/refinement between levels;
(ii) orthogonality—the possibility to model concurrent or independent submachines within one state machine;
(iii) broadcast communication—the possibility to synchronize multiple concurrent submachines via events. Each internal (produced by the system) of external (produced by the environment) event is instantaneously broadcasted.

statecharts = FSM + Abstraction + Orthogonality + Broadcast-communication

Some state-oriented approaches (e.g., Petri Nets) associate a transition with the execution of one concrete activity (or a group of activities). On the contrary, with statecharts we associate a state transition with the occurrence of a triggering event (or combinations of events) allowing for a *deferred activity binding*. Thanks to the deferred binding, at design-time, the process scenario can be seen as a sequence of events; the concrete activities that will produce these events can be selected or invented in run-time. The process enactment can be seen as a dynamic selection of activities to produce some outcomes (events) that make the process progress towards its (desired) final state.

Visual notation. In the statechart notation, states are depicted with rectangular boxes with rounded corners. Figure 3.2 illustrates a high level diagram for our flood management process example. The substate–superstate relation is depicted by boxes encapsulation. *Activation of Crisis Centers* and *EPSF* are exclusive substates of the *Flood Emergency* state: when in the *Flood Emergency* state, the case can be either in one or in the other of these substates. While entering the *Flood Emergency* state for the first time, the *Activation of Crisis Centers* substate is entered "by default"—this is depicted by the arrow with a black circle pointing at this substate.

Figure 3.3 shows a detailed diagram of the *EPSF* state from Fig. 3.2. The areas separated by the dashed lines represent the concurrent substates of their *EPSF* superstate: when in the *EPSF* state, the case is simultaneously in eight concurrent substates. Each of them can be seen as a separate statechart with its own state hierarchy. Thus,

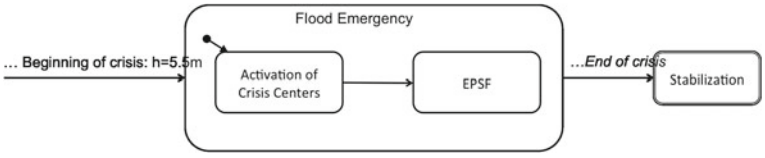

Fig. 3.2 High-level view of the Flood management process

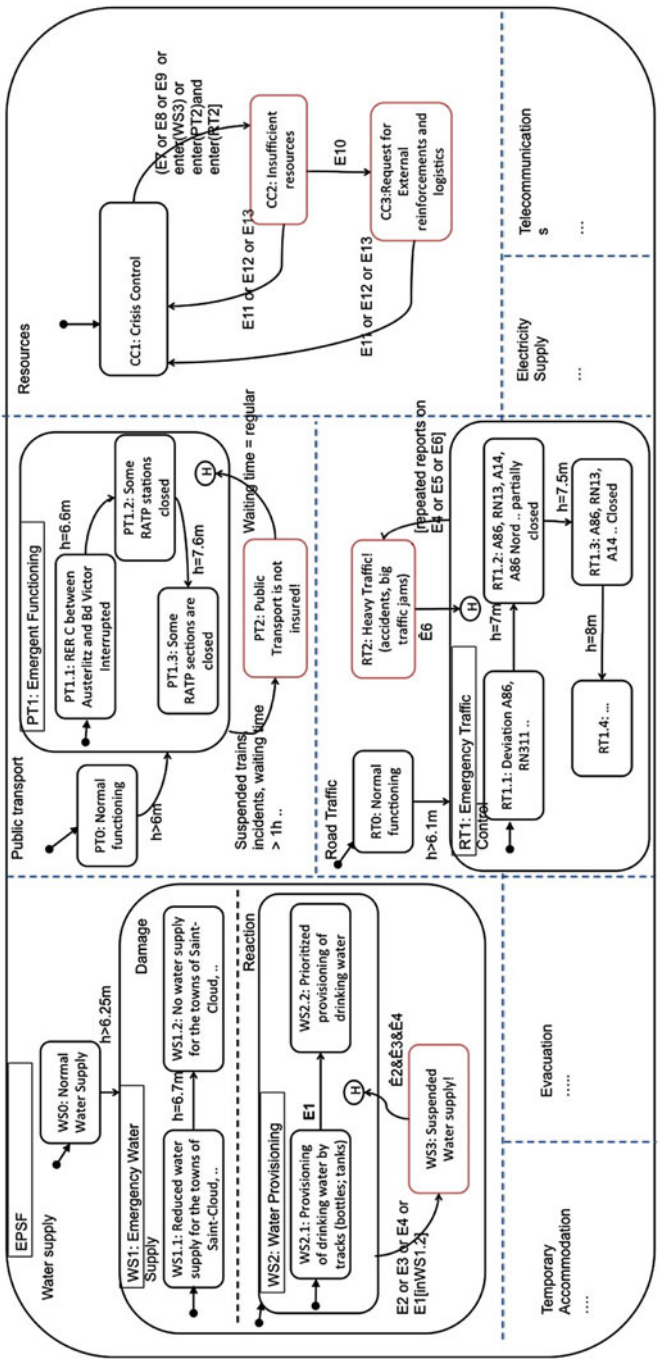

Fig. 3.3 Statechart diagram specifying the crisis management process once the EPSF is activated

the introduction of concurrent substates is a convenient mechanism to specify logically different areas of the case management (Public Transport management, Water Supply management, Road Traffic management etc.).

The set of active states of all concurrent substates is called the *active configuration* of a statechart. It replaces the term of "current state" in conventional (flat) FSM.

The transition that terminates with a circle with "H" stands for "entering the state by history". The transition from *PT1* to *PT2* in Fig. 3.3, for example, specifies that once the case recovers from the *PT2: Public Transport is Not insured* state and re-enters the *PT1:Emergent Functioning* state—the last active configuration of the latter is selected (and not the default one).

The transitions between states in statecharts are depicted by arrows labeled with expressions that specify the triggering events and (optionally) the actions that are carried out while the transition is triggered. In our example, the triggering events mostly represent external and internal events.

More details on the statecharts notation can be found in [23]. The semantics of statecharts for CMP will be presented in more details in Sect. 3.5.

Execution of statecharts specifications. The operational semantics of statecharts was originally implemented in the STATEMATE system and described in [21, 31]. The statecharts formalism was also adopted by the UML community in the form of UML statemachine diagrams [46].

Rhapsody [20] (now IBM Rational Rhapsody) and open source YAKINDU Statechart Tools (SCT) are examples of statecharts modeling environments, where the statecharts specifications can be created and executed in an intuitive and interactive way.

3.5 Statecharts Semantics for Case Management Processes

As explained above, we adopt the formalism of statecharts for the specification of case management processes (Sect. 3.5.1). We also propose some extensions of statecharts in Sect. 3.5.2.

We create the statecharts specification for the flood management process based on the description provided by the Emergency Plan Specialized on Floods (EPSF) and on some practical knowledge about resource management during floods. The resulting diagrams are shown in Figs. 3.2 and 3.3.

We start with a high-level view of the process described by two states—*Flood Emergency* and *Stabilization*—and transitions between them (Fig. 3.2). The *Flood Emergency* state is entered when the water level at Austerlitz Bridge raises above 5.5 m. It contains two substates: *Activation of Crisis Centers* and *EPSF*. The transition to *Stabilization* state is triggered once specific conditions identified as "*end of crisis*" are met.

The diagram in Fig. 3.3 specifies the main areas of crisis management as concurrent substates of the *EPSF* state. For the purpose of this work, we show only a few of

these substates in detail: *Water supply*, *Public transport*, *Road traffic* and *Resources*. This model can be refined providing further details on the crisis management scenarios and operation procedures.

3.5.1 Statecharts Semantics for CMP Specification

Below, we explain how the following concepts defined by the statecharts formalism [23] can be adopted for the specification of CMP:

- State, state hierarchy and state decomposition;
- Abstraction and refinement;
- AND, OR and basic states;
- Entering a state by default and by history;
- State configuration;
- Internal, external and triggering events;
- Activity;
- Broadcast communication;
- Inter-level transition.

State, state hierarchy and state decomposition. A CMP state can be seen as a specific situation in the case management process that requires reaction.

On the abstract level, states can be compared to business milestones. The definition of the right set of states for the process is subjective: it reflects our current understanding of the process and evolves over time. In this work, the states of the flood crisis management process are characterized by the level of water h. These states represent the critical points for different management areas defined by EPSF (Sect. 3.2).

While *being in a given state*, some work has to be done in order to maintain this state or in order to leave this state and enter another state. Note that statecharts do not specify how exactly this work will be performed or which activities will be executed and in which order. Another means for modeling activities is needed: statecharts, for example, can be complemented with activity charts [23]. In this paper, we do not discuss activity modeling in detail.

In Fig. 3.3, three states *RT0*, *RT1* and *RT2* specify the main phases of the road traffic control after the emergency plan (EPSF) is triggered.

- *RT0: Normal functioning* is the default state upon triggering the EPSF. The water level of 5.5 m does not disrupt the road infrastructure of the region and normal functioning is maintained.
- *RT1: Emergency traffic control*—this state is attained at 6.1 m; here the flood is affecting the road traffic. Specific measures must be continuously taken in this state in order to maintain road safety.

- *RT2: Heavy Traffic!*—this state is reached when the road traffic degrades (due to accidents, traffic jams) to the point where the crisis management itself becomes compromised (e.g., the rescue teams cannot arrive to the endangered areas, etc.).

A state *s* consists of a (possibly empty) *hierarchy* of substates, representing (possibly concurrent) state machines. These substates provide details about their parent state (or superstate).

In Fig. 3.3, four different substates (from *RT1.1* to *RT1.4*) are defined based on the flood severity: upon entering each of these states, the city executes some scenario: deploying equipment, marking deviations, blocking roads, informing drivers, etc. Each substate belongs to one superstate (its surrounding state) that is also its nearest ancestor in the state hierarchy. We call the relations between the superstate and its substates abstraction/refinement relations.

Abstraction and refinement. State abstraction consists in clustering states into a superstate according to some similarity criteria. This mechanism allows one to describe the problem at multiple abstraction levels, hiding or introducing details when necessary. Refinement is the opposite of abstraction, it consists in decomposing a state into substates according to some discrimination criteria.

More formally, *refinement* is a XOR decomposition of a state, where being in a superstate means being in exactly one of its (exclusive) component substates.

One substate can be marked as default so that this state is visited each time its parent state is entered.

Public Transport Emergent Functioning state (PT1) in Fig. 3.3 is specified with three exclusive substates corresponding to three different management scenarios that are activated based on the water level *h*. *PT1.1* is the default scenario.

From a visualization standpoint, clustering states allows for a very economical representation. It avoids duplicating transitions and the model logical structure appears clearly.

The AND decomposition results in the specification of orthogonal (or concurrent) components of the parent state. The AND decomposition models the situation when being in the state means being in one of the combinations of its components. All possible combinations make an orthogonal product.

AND, OR and basic states. The statecharts formalism defines three types of states: AND, OR and basic states. The AND-state is a state that contains two or more orthogonal substates; the OR-state is a state that contains one or more exclusive substates. A state is basic if it does not have any substates.

Consider the *Water Supply* in Fig. 3.3: *WS1:Emergency water supply* is an AND-state that contains two concurrent substates *Damage* and *Reaction*. These substates model the damage due to the flood and the reaction to it, i.e., emergency water provisioning. Once *WS1* is entered both of its concurrent substates are activated.

The *Damage* state is an OR-state; it contains two exclusive substates that specify its details: *WS1.1*, where the water supply of some towns is reduced; *WS1.2*, where the water supply is totally disrupted.

The *Reaction* state is an OR-state; it contains two exclusive substates: *WS2: Water provisioning* and *WS3: Suspended Water supply!*

Once *WS1* is entered,*WS1.1, WS2* and its substate *WS2.1* are entered by default.

Emergent water provisioning (WS2) defines specific measures to provide areas with drinking water: normal provisioning (WS2.1) and prioritized provisioning (WS2.2) in case of limited stock of drinking water (event *E1*). The state *WS3: Suspended Water Supply!* has no substates—it is a basic state. It refers to the situation when the emergent water provisioning can no longer be guaranteed. This, for instance, can result from severe road conditions or insufficient stock of bottled drinking water while no other supply is available (i.e., when the case is in *WS1.2* state). This is indicated by the transition label: *E2 or E3 or E4 or E1 [in WS1.2]].*

Entering a state by default and by history. The default indicator is used to identify which substate will be visited when its parent state is entered. Alternatively, in many cases it can be useful to enter the superstate by history, i.e., to enter its most recently visited substates (or configuration of substates). The examples include the transition from *WS3* back to *WS2* in Fig. 3.3: once the problem is solved, the emergency water supply is restored at its latest visited substate (which is not necessarily *WS2.1*).

State configuration. Compared to conventional (flat) FSM, in hierarchical state machines depicted by statecharts, multiple states can be activated at the same time. Statecharts define the term *configuration* (of a state or of a system):

The *active configuration* of a state s is the set of basic substates of s that are activated at the current moment. Intuitively, active configuration replaces the conventional term of current state defined in FSM.

For the state WS1 in Fig. 3.3, consider that $h = 7\,\text{m}$ and the water provisioning functions normally; this would correspond to the following active configuration of *WS1*: $cf(WS1) = WS1.2, WS2.1$

The sequence of active configurations resulting from the execution of the statechart specification represents a *trace* of the CMP.

Internal, external and triggering events. *Internal events* are produced by the system (Fig. 3.1); they are the results of carried out activities.

In Fig. 3.3, event *E1* specifies an *insufficient stock of drinking water*. It is an internal event that can result from the water distribution activity or can be generated by some other activity like stock verification.

External events are produced by the environment (context) of the case. The case context consists of various objects that influence the case and affect its handling (Fig. 3.1). Water level, weather forecast, current situation on the roads, incident reports are examples of contextual parameters *sensed* by a system during a flood.

In Fig. 3.3, *WS0–WS1* or *WS1.1–WS1.2* state transitions are taken if a certain value of h (water level) is reached or exceeded. We consider that the change in the water level is an external (contextual) event.

The *triggering event e[c]* (interpreted as *e occurs and c holds*) of a transition t is an event that must occur in order for t to take place. Here $e \in E$ is the event (or a logical combination of events) that triggers the transition; $c \in C$ is a condition that needs to be true for the transition to be taken when e occurs.

In Fig. 3.3, the triggering event for the transition from *WS2* to *WS3* is described by an expression: *E2 or E4 or E1[in(WS1.2)]*. This transition is taken if no more tracks

for transporting bottled water are available or if the road to the concerned area is blocked or if the stock of drinking water on place is insufficient while some towns no longer have regular water supply. The operational information about the resources or traffic conditions corresponds to contextual or internal events. Condition *in(WS1.2)* specifies that *WS1.2* substate is active.

To specify some work to be done, statecharts use the concept of activity.

Activity. The statecharts formalism defines *state-dependent activities* that are linked directly to a state *s* and can be carried out *throughout* or *within s*. In the first case, an activity starts when entering the state *s* and terminates when leaving it. In the second case, an activity starts when entering *s*; when exiting *s*, if the activity it is not terminated yet, it is stopped by the system.

This is a valid interpretation for a case management process too: in our example, upon entering the state *WS2.1* (in Fig. 3.3) an activity for water provisioning must be started and must continue within this state. Upon entering *PT1*, activities for closing the concerned stations must be carried out *throughout* this state.

Relations between activities and states defined by statecharts can be characterized as mandatory: if activity *A* is linked to state *s* by a *throughout* or a *within* relation it must be carried out at this state. Therefore, each state *s* can be associated with a (possibly empty) set of mandatory activities.

Broadcast communication. Broadcasting allows for communication and synchronization between concurrent sub-machines. According to statecharts, both internal and external events are *broadcasted*, meaning that one single event can trigger transitions at multiple orthogonal substates.

Broadcast communication allows for coordination between different management areas in CMP. For example, *Road Traffic* and *Resources* substates of *EPSF* can both react to an (external) event reporting on the hard traffic in a particular road section. Along those lines, blocked roads or insufficient water supply (internal events) may trigger evacuation of people and facilities from the concerned area.

Inter-level transitions. The transitions that cross state boundaries are called inter-level transitions in statecharts. Their purpose is to model the interruptions or the situations when the process has to react, no matter its state or the activity it performs. For example, if the *End of crisis* event occurs (Fig. 3.3), no matter what the configuration of *EPSF* substates is and what activities are executed in all its areas, they will be terminated and the new state (presumably *Stabilization*) will be entered by the process.

3.5.2 Adaptation and Extension of the Statecharts Formalism for CMP Specification

Below, we discuss the specific features of CMP that cannot be captured by the original statecharts concepts and we therefore propose adaptations and extensions to the statecharts formalism.

What kinds of extensions are needed? Why? Despite the similarities identified in Sect. 3.4, there exists a number of characteristics that makes a CMP significantly different from a conventional reactive system:

1. A CMP has a goal. In reaction to given stimuli in a given situation, the case manager searches for scenarios that could steer the case towards its goal. Thus, compared to a reactive system where the next state (or active configuration) is defined by its current state and a given situation, the next state in CMP is also defined by the process goal.
2. CMP is a knowledge-intensive process where decisions (e.g., scenario planning, task assignment) are typically made by the case manager. As a consequence:
3. CMSS can be compared to business-intelligence systems (BI) rather than automated control systems: for the latter, once the preconditions are satisfied for an activity *a*, *a* is *automatically executed*. For CMSS, once the preconditions of *a* are satisfied, *a* is *enabled for execution* and (unless explicitly stated as *mandatory*) the case manager decides weather it will be carried out or not.
4. Whereas some events are relevant only immediately after they occur (e.g., button pressed), other events, once they occur, remain relevant or valid for some period of time or for the whole execution of a CMP (e.g., document received; permission granted).

In order to faithfully represent the complexity of a CMP, we propose to extend the statecharts formalism with the following concepts:

1. Final configuration;
2. Path, path selection and path reinforcement;
3. Relevance and validity interval for events;
4. Mandatory versus optional activities.

We briefly describe these concepts in the remainder of this section.

Final configuration. Similarly to [51], we express the process *goal* in terms of "target state" (or configuration) that the state transition system has to reach or to maintain; strategies (possible scenarios) for achieving this goal can be seen as sequences of states to visit (or state transitions to fire) between the "current" and the "target" states.

By analogy with the active configuration defined by statecharts, we define a final configuration for CMP:

The *final configuration* of a state *s* is the set of basic substates of *s* that we want to enter and/or maintain upon the CMP termination.

In our example, the goal of the process is to support the areas affected by the flood and to protect the population from the flood consequences. For the statechart

in Fig. 3.3, any configuration where the critical states *WS3, PT2, RT2, CC2, CC3* are not active can be considered as a final configuration (i.e., it should be maintained until the crisis is over).

Path, path selection and path reinforcement. We define a *path* in statecharts as any sequence of active configurations that terminates with the final configuration.

In our example, if one of the concurrent submachines has entered a critical state, the path is the sequence of configurations that would lead this submachine back to one of its non-critical states. For example, if the state *CC2: Insufficient resources* of the *Resources* substate is active, there are two paths for this submachine to bring it back to CC1: $CC2 \longrightarrow CC1$ or $CC2 \longrightarrow CC3 \longrightarrow CC1$.

In any given active configuration, a path towards the final configuration can be calculated. The optimal path can be selected using some criterion (e.g., the cheapest path, the shortest path, the path with the highest probability to be realized).

Consider the optimal path p from the current active configuration to the final configuration. *Enforcing path p* means executing some activities in order to enable and then to take some state transition t that will lead to the next active configuration in p.

Consider that the state *CC2: Insufficient resources* of the *Resources* submachine is active and that the path $CC2 \longrightarrow CC1$ is the optimal path. Enforcing this path means here enabling the transition from *CC2* to *CC1*. This transition will be taken if at least one of the events *E11* (added manpower), *E12* (added supplies) or *E13* (added equipment) occurs. We can enforce the path by mobilizing volunteers or relocating supplies, manpower or equipment, considering that these activities can generate E11, E12 or E13 as a result.

Relevance and validity interval for events. To take some transition t, the execution of process activities and/or observation of contextual events can be required.

Most process formalisms including statecharts define a triggering event as a single event, or a group of events that occur simultaneously and instantly trigger a state transition. In particular, statecharts specify that *events are only available in the step directly succeeding their generation* [19]. We call such events **instantaneous** events and distinguish them from **continuous** events that, once observed, remain valid and can be reacted upon asynchronously, during multiple steps.

We define the *validity interval tv* for event e as a period of time between the moment when this event is first observed and the moment when it becomes irrelevant for the process.

To define the validity interval for an event e, we associate it with the time the system resides in some state or with the occurrence of another event $\hat{e}$ that cancels e:

If the validity interval tv of event e is some state s: $tv(e) = s$, this means that, after being sensed for the first time in s (or one of its substates), e will be valid until the system leaves s. The higher the state in the state hierarchy, the longer the validity interval.

If the validity interval tv of event e is some event $\hat{e}$: $tv(e) = \hat{e}$, this means that, after being sensed, e will be valid until $\hat{e}$ occurs.

For example, the *approval received* event for starting the evacuation procedure (not shown in the statechart in Fig. 3.3) is valid as long as the *Flood Emergency* is active; *E6* (traffic jam) event is valid until $\hat{E6}$ (fluid traffic) event is received.

If no validity interval is specified for an event, this event is an *instantaneous* event.

We extend the definition of triggering event as follows: *the triggering event e[c]* of a transition t is an event that must occur for t to take place. Here $e \in E$ is a combination of events and/or absence of events observed during some period of time (validity interval) that triggers the transition; $c \in C$ is a condition that needs to be true in order the transition to be taken when e occurs.

This definition allows us to take into account not only immediate events but also relevant events observed in the past (email received, approval obtained, etc.).

Mandatory versus optional activities. In the statecharts formalism, activities can be considered as *state-dependent* [56] (i.e., each state s is associated with a list of activities to be carried out in this state). These activities are also *mandatory*: they are automatically executed once their preconditions are met.

To relax the coupling between *a situation* and *a reaction* and to allow for more flexibility in process execution, we propose to define *state-independent* activities for statecharts—activities, that can be executed in any configuration if their preconditions are met (unless explicitly stated otherwise).

Thus, each state s can be associated with two (possibly empty) sets of activities: the set of *mandatory* activities that must be carried out in s and are state-dependent and the set of optional or *enabled* activities that are defined dynamically, based on the statechart status. Optional activities can be executed *within* or *throughout* the state in order to ensure the right progression of the case towards its goal.

The specification of activities is beyond the scope of this chapter.

3.6 Perspectives and Roadmap for Future Research

The benefits of the proposed formalism are numerous for CMP: executable statecharts specifications allow for interactive design, simulation-based testing and simulation-based recommendations. In the future, these features could be integrated as a part of CMP-supporting PAIS in order to provide intelligent decision-support functionalities for case managers. To conclude this chapter we discuss these perspectives and outline the directions for future work.

3.6.1 Design and Simulation-Based Testing

Interactive design of CMP. A statecharts specification can be created based on some a priori knowledge about the CMP (e.g., norms, regulations, best practices, etc.) (Fig. 3.4a). Thanks to the concept of hierarchical state, this model can be extended

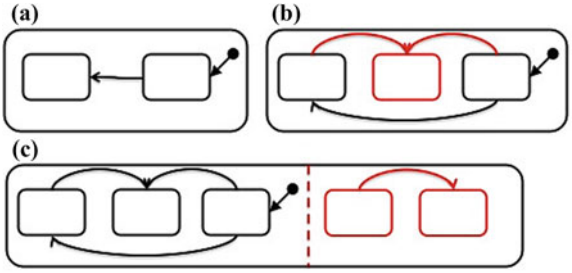

Fig. 3.4 Incremental design of CMP

and refined by integrating the experience of the case manager: new states and state transitions can be specified reflecting new situations and the way to deal with them (Fig. 3.4b); concurrent substates can be added in order to increase the scope of the process (Fig. 3.4c). IBM Rational Rhapsody[10]) and open source YAKINDU Statechart Tools (SCT)[11] are examples of statecharts modeling environments, where the statecharts specifications can be created and executed in an intuitive and interactive way. However, creating a detailed statecharts specification for a real-life CMP is a challenging task, as explained below.

Using clustering techniques for statecharts improvement. Although statecharts have been designed to represent states in a hierarchical way, this formalism does not specify how states should be organized into abstraction levels. In most cases this clustering of states is performed manually by a process designer.

Clustering algorithms gather entities (e.g., the states of a state machine) into clusters according to some similarity criteria that can include complex sets of parameters. Formal Concept Analysis (FCA) [17] is a well-known clustering technique that is successfully used in many areas including knowledge discovery, representation and sharing [39]. In FCA, the obtained clusters are organised into a lattice using generalisation and specialisation relationships, which could be used to identify state hierarchy in statecharts. A significant advantage of FCA is that the resulting clusters may overlap, whereas many traditional clustering techniques build partitions.

FCA can therefore be used for clustering states and helping to define the hierarchical structure within a statecharts specification. Various attributes may be chosen to describe states: pre-conditions, post-conditions, contextual parameters, and any combination of them. As a result, each FCA concept (cluster) is explicitly labelled by the set of attributes that characterize the objects of the cluster. This can be considered as a starting point for further model analysis and improvement: detection of "missing" states or state transitions, identification of "similar" states or activities etc.

Simulation of CMP specifications. Statecharts combine an intuitive and concise visual notation with precise semantics. Thanks to these semantics, the statecharts

[10] http://www-03.ibm.com/software/products/en/ratirhapfami.

[11] http://statecharts.org/index.html.

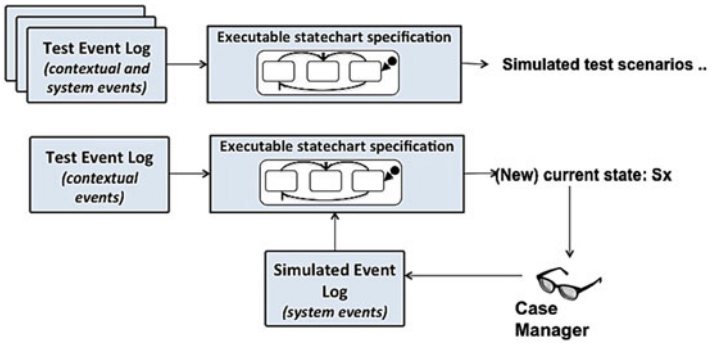

Fig. 3.5 Simulation-based testing of CMP scenarios

specifications can already be simulated at the early design stages, providing an instant visual feedback. At the later design stages, they can serve as a basis for simulation-based testing as explained below.

A statecharts specification can be executed with a test event log (i.e., a pre-recorded sequence of events defining some flood development scenario) allowing for the simulation and testing of various handling scenarios. Two simulation modes can be defined:

- A fully automated mode (Fig. 3.5a), where the statecharts specification is executed with a test event log that includes both contextual events (e.g., raise of water level, traffic jam) and system events (e.g., successful deployment of equipment, empty stock of drinking water). The events from the event log are processed by a statecharts simulation environment[12] triggering the state transitions. The simulation result is a sequence of visited states.
- In an interactive mode (Fig. 3.5b), the test event log contains only contextual (external) events and emulates the environment. A case manager reacts to external events by executing enabled activities (e.g., deploying equipment, making task assignments)—these activities represent the steps of case handling scenarios.

Similarly to computer simulator games, the interactive statecharts simulation is an iterative process, where the case response is simulated after each step taken by the case manager: pre-recorded external events and internal events resulting from the case manager's decisions trigger state transitions in statecharts specification and once the (new) current state **Sx** is entered a new step starts. The simulation result is the sequence of visited states, executed activities and received events.

In the case of a crisis management process, multiple scenarios can be "played" automatically or interactively and used as a basis for trainings, drills and improvement of formal operation procedures (e.g., procedures described by EPSF).

[12]Development of modeling and simulation environment for CMP will be addressed in our future work.

Conversely, possible case development scenarios can be calculated as sequences of events acceptable by the state machine representing the CMP. This could help to analyse the process and reveal scenarios that were not considered before.

3.6.2 Simulation-Based Recommendations

Gartner's Hype Cycle for Emerging Technologies report provides a cross-industry perspective on technologies and trends, with an assessment of their maturity, adoption and business benefit. According to the reports from 2013 and 2014,[13] Predictive Analytics technologies have already reached their plateau of productivity and are currently becoming the mainstream technology, whereas Complex-Event Processing (CEP), Big Data and Content Analytics are currently rolling down from their peak of inflated expectations and will reach their maturity (the plateau) in 5 to 10 years. This makes run-time situation analysis and recommendations for case managers the next challenge for the CMP-supporting PAIS.

Some recommendation systems supporting process modeling and process management are presented in the literature [29, 50]. Process mining is a widely recognised technique for predicting a best process scenario based on the analysis of past execution logs [4]. The approach reported in [8] uses constraint-based programming to generate recommendations on process execution strategies.

An example of CMP solution integrating intelligent support for the case manager is reported in [26]. Here the authors introduce the concept of User-Trained Agent (UTA), which recommends the best next actions based on the actions taken by the case managers in previous similar situations. The proposed recommendation technique is based on pattern recognition and is integrated as a part of ISIS Papyrus platform.

Whereas all the approaches for recommendations mentioned above are based on "past experience" or process logs, we propose an alternative technique that is based on *the execution of a CMP specification in the simulated process environment*:

In our vision, the "a priory" statecharts specification of a CMP can be analysed using graph theory algorithms. The objective of the analysis is to search and optimize a path from some current state of the statecharts model to its target state, representing the goal of the process. As a result, the "best next state to visit", "best next transition to fire" and, consequently, "possible activities to execute" are recommended to the case manager. The main advantage of this analysis is that recommendations can be provided based on:

- our current knowledge about the process represented by its executable statecharts specification and
- our current knowledge about the process environment, represented by a real-time event buffer (or event log).

[13] Gartner, http://www.gartner.com/newsroom/id/2575515, http://www.gartner.com/newsroom/id/2819918.

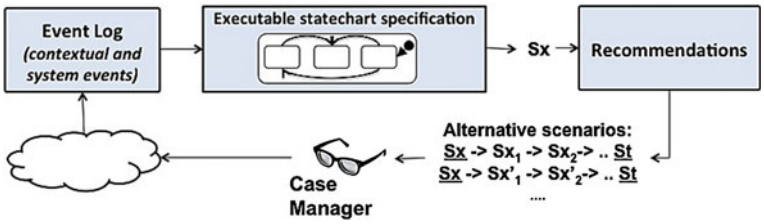

Fig. 3.6 Run-time recommendations on the CMP activity planning

No "past experience" represented by a log of the past process executions is required. This makes "cold starts" possible.

A statecharts specification could be initialized and then executed using a *real-time event log* (i.e., where both contextual (external) and system (internal) events occur in real time). Given a current state **Sx** of the statecharts and the desired (target) state **St**, possible case management scenarios can be calculated as alternative *paths* from **Sx** to **St** on the statecharts diagram. Each scenario can be seen as a sequence of "correct" state transitions resulting from the execution of corresponding activities (Fig. 3.6).

The alternative scenarios and activities that need to be executed in order to reinforce these scenarios could then be recommended to the case manager.

The integration of CMP executable specifications and analysis tools as a part of CMP-supporting PAIS could provide an intelligent support for case managers, as explained below.

3.6.3 Enhancing the CMP-supporting PAIS with Recommendations for Agile Activity Planning

Figure 3.7 illustrates our *vision* of the intelligent CMSS introduced in our earlier works [48, 49]. We describe below the main components of this system: Dynamic context manager (DCM), Navigation manager (NM), Activity/Resource repository, Log and History.

The role of *the Dynamic context manager* is to select, measure and monitor relevant contextual variables of the CMP. Internal and external events are collected and stored in the *Event log*. The *Activity/Resource repository* stores the definitions of activities that can be performed during the case handling and resources that can be used. The *Log and History* component registers the ongoing process scenario (i.e., the sequence of executed activities, received events and visited states).

The *Navigation Manager* is the "heart" of the system, it provides intelligent support for the case manager by recommending the best scenario(s) for handling the case. It contains the *Executable statecharts specification* of the CMP and the *Recommendation* component that uses graph theory, process mining and clustering algorithms in order to provide recommendations for the case manager.

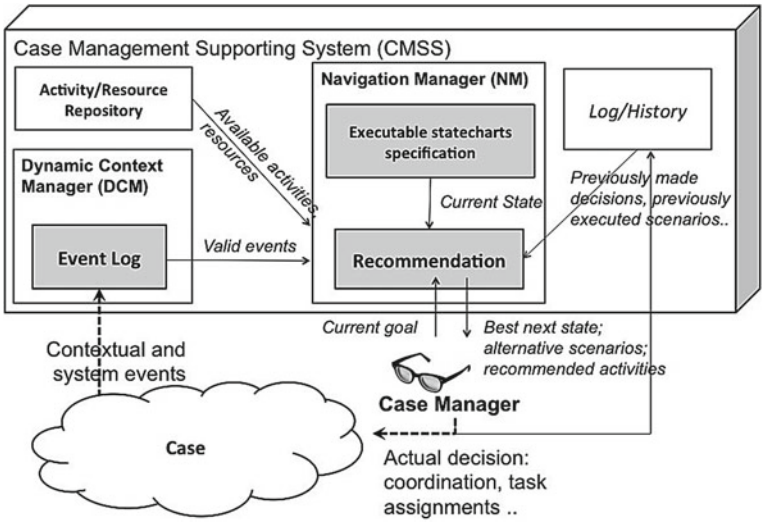

Fig. 3.7 Intelligent CMSS. The Navigation Manager provides recommendations about the best scenario based on the current state of the process and the list of valid events

The *Executable statecharts specification* of CMP models the *behavior* of the SUD and its Environment. It can be executed with the collected real-time events. Possible case management scenarios are described by the sequences of states of the statecharts model that lead from the current state to some target state that represents the CMP objective.

The *Recommendation* component can provide the case manager with an insight about how the situation might develop and about the possible strategies (paths in statecharts, activities, groups of activities to carry out) to bring the situation under control. The recommendation mechanism uses the Activity/Resource repository to define the list of activities enabled at a given situation identified with the current configuration (state) of the statecharts model. Since activities are independent from states, new activities can be added to the Activity/Resource repository at run time and further used by the Recommendation component without needing to change the model.

The intelligent CMSS sketched above will be grounded on the statecharts specifications enabling incremental interactive process design, simulation-based testing and recommendations. According to the statecharts formalism, a case management process is represented by a hierarchical state machine, where the process scenario can be seen as *a dynamic choice of activities with an objective to trigger a "good" state transition that would move the case from its "current state" towards its "target state" representing the case management goal.*

The development of a prototype of the intelligent CMSS is our future work.

References

1. van der Aalst, W.: The application of petri nets to workflow management. J. Circ. Syst. Comput. **8**(01), 21–66 (1998)
2. van der Aalst, W.: Process-aware information systems: lessons to be learned from process mining. In: Jensen, K., van der Aalst, W. (eds.) Transactions on Petri Nets and Other Models of Concurrency II. LNCS, vol. 5460, pp. 1–26. Springer (2009)
3. van der Aalst, W., Ter Hofstede, A.H.: Yawl: yet another workflow language. Inf. Syst. **30**(4), 245–275 (2005)
4. van der Aalst, W., Weijters, A.: Process mining: a research agenda. Comput. Ind. **53**(3), 231–244 (2004)
5. van der Aalst, W., Pesic, M., Schonenberg, H.: Declarative workflows: balancing between flexibility and support. Comput. Sci. Res. Dev. **23**(2), 99–113 (2009)
6. van der Aalst, W., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. Data Knowl. Eng. **53**(2), 129–162 (2005)
7. Bajec, M., Krisper, M.: A methodology and tool support for managing business rules in organisations. Inf. Syst. **30**(6), 423–443 (2005)
8. Barba, I., Weber, B., Del Valle, C.: Supporting the optimized execution of business processes through recommendations. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) Business Process Management Workshops. LNBIP, vol. 99, pp. 135–140. Springer, Berlin (2012)
9. Bider, I.: Towards a non-workflow theory of business processes. In: La Rosa, M., Soffer, P. (eds.) Business Process Management Workshops. LNBIP, vol. 132, pp. 1–2. Springer, Berlin (2013)
10. Bubenko, J., Rolland, C., Loucopoulos, P., DeAntonellis, V.: Facilitating fuzzy to formal requirements modelling. In: Proceedings of the First International Conference on Requirements Engineering, 1994, pp. 154–157. IEEE (1994)
11. Cauvet, C.: Modélisation des processus d'ingénierie des systèmes d'information. Encyclopédie de l'Informatique et des Systèmes d'Information, pp. 1412–1425 (2006)
12. Davenport, T.: Thinking for a Living: How to Get Better Performances and Results from Knowledge Workers. Harvard Business Press (2005)
13. Dijkman, R., Dumas, M., Garca-Bauelos, L.: Graph matching algorithms for business process model similarity search. In: Dayal, U., Eder, J., Koehler, J., Reijers, H. (eds.) Business Process Management. LNCS, vol. 5701, pp. 48–63. Springer, Berlin (2009)
14. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Inf. Softw. Technol. **50**(12), 1281–1294 (2008)
15. Dowson, M.: Iteration in the software process; review of the 3rd international software process workshop. In: Proceedings of the 9th International Conference on Software Engineering, ICSE '87, pp. 36–41. IEEE Computer Society Press, Los Alamitos, CA, USA (1987)
16. Dumas, M., van der Aalst, W.M., Ter Hofstede, A.H.: Process-Aware Information Systems: Bridging People and Software Through Process Technology. Wiley (2005)
17. Ganter, B., Wille, R., Wille, R.: Formal Concept Analysis, vol. 284. Springer, Berlin (1999)
18. Groefsema, H., Bucur, D.: A survey of formal business process verification: from soundness to variability. In: Proceedings of International Symposium on Business Modeling and Software Design (BMSD) (2013)
19. Harel, D.: Statecharts: a visual formalism for complex systems. Sci. Comput. Program. **8**(3), 231–274 (1987)
20. Harel, D., Gery, E.: Executable object modeling with statecharts. In: Proceedings of the 18th International Conference on Software Engineering, ICSE '96, pp. 246–257. IEEE Computer Society, Washington, DC, USA (1996)
21. Harel, D., Naamad, A.: The statemate semantics of statecharts. ACM Trans. Softw. Eng. Methodol. **5**(4), 293–333 (1996)
22. Harel, D., Pnueli, A.: On the development of reactive systems. Springer (1985)
23. Harel, D., Politi, M.: Modeling Reactive Systems with Statecharts: The STATEMATE Approach. McGraw-Hill, Inc. (1998)

24. Hull, R., Damaggio, E., De Masellis, R., Fournier, F., Gupta, M., Heath, III, F.T., Hobson, S., Linehan, M., Maradugu, S., Nigam, A., Sukaviriya, P.N., Vaculin, R.: Business artifacts with guard-stage-milestone lifecycles: managing artifact interactions with conditions and events. In: Proceedings of the 5th ACM International Conference on Distributed Event-Based System, DEBS '11, pp. 51–62. ACM, New York, NY, USA (2011)
25. Kemsley, S.: The changing nature of work: from structured to unstructured, from controlled to social. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) Business Process Management. LNCS, vol. 6896, pp. 2–2. Springer, Berlin (2011)
26. Kim, T.T.T., Ruhsam, C., Pucher, M.J., Kobler, M., Mendling, J.: Towards a pattern recognition approach for transferring knowledge in ACM. In: Enterprise Distributed Object Computing Conference Workshops and Demonstrations (EDOCW), pp. 134–138 (2014)
27. Kirsch-Pinheiro, M., Rychkova, I.: Dynamic context modeling for agile case management. In: Demey, Y., Panetto, H. (eds.) On the Move to Meaningful Internet Systems: OTM 2013 Workshops. LNCS, vol. 8186, pp. 144–154. Springer, Berlin (2013)
28. Koehler, J., Tirenni, G., Kumaran, S.: From business process model to consistent implementation: a case for formal verification methods. In: Enterprise Distributed Object Computing Conference, 2002, EDOC '02. Proceedings, pp. 96–106 (2002)
29. Koschmider, A., Oberweis, A.: Designing business processes with a recommendation-based editor. In: Brocke, J., Rosemann, M. (eds.) Handbook on Business Process Management 1. International Handbooks on Information Systems, pp. 299–312. Springer, Berlin (2010)
30. van Lamsweerde, A.: Goal-oriented requirements engineering: a guided tour. In: 2001. Proceedings. Fifth IEEE International Symposium on Requirements Engineering, pp. 249–262 (2001)
31. Mikk, E., Lakhnech, Y., Petersohn, C., Siegel, M.: On formal semantics of statecharts as supported by statemate. In: Workshop, Ilkleym, vol. 14, p. 15 (1997)
32. Murata, T.: Petri nets: properties, analysis and applications. Proc. IEEE **77**(4), 541–580 (1989)
33. Nurcan, S., Edme, M.H.: Intention-driven modeling for flexible workflow applications. Softw. Process: Improv. Pract. **10**(4), 363–377 (2005)
34. OMG: Case management process modeling (CMPM) request for proposal. http://www.omg.org/cgi-bin/doc?bmi/09-09-23 (2009)
35. OMG: Business process model and notation (BPMN). http://www.omg.org/spec (2011)
36. OMG: Case management model and notation. http://www.omg.org/spec/CMMN/1.0/PDF/ (2014). document number formal/2014-05-05
37. Pesic, M., Schonenberg, H., van der Aalst, W.: Declare: full support for loosely-structured processes. In: Enterprise Distributed Object Computing Conference, 2007, EDOC 2007. 11th IEEE International, pp. 287–287. IEEE (2007)
38. Plotkin, G.: A structural approach to operational semantics (1981)
39. Poelmans, J., Elzinga, P., Viaene, S., Dedene, G.: Formal concept analysis in knowledge discovery: a survey. In: Conceptual Structures: From Information to Intelligence, pp. 139–153. Springer (2010)
40. Pohl, K., Weidenhaupt, K.: A contextual approach for process-integrated tools. In: Jazayeri, M., Schauer, H. (eds.) Software Engineering ESEC/FSE'97. LNCS, vol. 1301, pp. 176–192. Springer, Berlin (1997)
41. Pucher, M.: The elements of adaptive case management. In: Mastering the Unpredictable, pp. 89–134 (2010)
42. Reijers, H.A., Limam, S., Van Der Aalst, W.: Product-based workflow design. J. Manag. Inf. Syst. **20**(1), 229–262 (2003)
43. Rolland, C., Prakash, N., Benjamen, A.: A multi-model view of process modelling. Requir. Eng. **4**(4), 169–187 (1999)
44. Rolland, C., Souveyet, C., Moreno, M.: An approach for defining ways-of-working. Information Systems **20**(4), 337–359 (1995)
45. Rosemann, M., van der Aalst, W.: A configurable reference modelling language. Inf. Syst. **32**(1), 1–23 (2007)

46. Rumbaugh, J., Jacobson, I., Booch, G.: Unified Modeling Language Reference Manual, 2nd edn. Pearson Higher Education (2004)
47. Rychkova, I.: Exploring the alloy operational semantics for case management process modeling. In: 2013 IEEE Seventh International Conference on Research Challenges in Information Science (RCIS), pp. 1–12 (2013)
48. Rychkova, I., Kirsch-Pinheiro, M., Le Grand, B.: Context-aware agile business process engine: foundations and architecture. In: Nurcan, S., Proper, H., Soffer, P., Krogstie, J., Schmidt, R., Halpin, T., Bider, I. (eds.) Enterprise, Business-Process and Information Systems Modeling. Lecture Notes in Business Information Processing, vol. 147, pp. 32–47. Springer, Berlin Heidelberg (2013)
49. Rychkova, I., Le Grand, B., Kirsch-Pinheiro, M.: Adaptive case management: supporting knowledge intensive processes with it systems. In: Fischer, L. (ed.) Empowering Knowledge Workers. BPM and Workflow Handbook Series. Future Strategies Inc. (2013)
50. Schonenberg, H., Weber, B., van Dongen, B., van der Aalst, W.: Supporting flexible processes through recommendations based on history. In: Dumas, M., Reichert, M., Shan, M.C. (eds.) Business Process Management. LNCS, vol. 5240, pp. 51–66. Springer, Berlin (2008)
51. Soffer, P., Yehezkel, T.: A state-based context-aware declarative process model. In: Enterprise, Business-Process and Information Systems Modeling, pp. 148–162. Springer (2011)
52. Swenson, K.: Mastering The Unpredictable: How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Do. Meghan-Kiffer Press (2010)
53. Swenson, K., Palmer, N., Manuel, A., Carlsen, S.: Empowering Knowledge Workers. BPM and Workflow Handbook Series. Future Strategies Inc. (2013)
54. Swenson, K., Palmer, N., Pucher, M., Manuel, A., Webster, C.: How Knowledge Workers Get Things Done. Future Strategies Inc. (2012)
55. Taylor, F.W.: The principles of scientific management. Harper (1914)
56. Wagner, F., Schmuki, R., Wagner, T., Wolstenholme, P.: Modeling software with finite state machines: a practical approach. CRC Press (2006)
57. Weske, M.: Business Process Management: Concepts, Languages, 2nd edn. Architectures. Springer, Berlin (2012)
58. Yu, E.S.: Towards modelling and reasoning support for early-phase requirements engineering. In: Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on. pp. 226–235. IEEE (1997)