UNIVERSITÀ DEGLI STUDI DI FIRENZE
Dipartimento di Ingegneria dell'Informazione

Dottorato di Ricerca in
Ingegneria Informatica, Sistemi e Telecomunicazioni
ING-INF/05

Multi-level meta-modeling architectures applied to eHealth

Fulvio Patara

Dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

in Informatics, Systems and Telecommunication Engineering

Ph.D. Coordinator	*Advisor*
Prof. Luigi Chisci	Prof. Enrico Vicario

XXVIII Ciclo – 2013-2016

To Maria Grazia

Acknowledgements

I would like to thank my advisor Prof. Enrico Vicario, for his guidance and advices during these three years. I would also like to thank Prof. Chris Nugent, for the opportunity given to spend a period of research at the Smart Environments Research Group (SERG) Laboratory of the Ulster University in Belfast, and for introducing me to the fascinating world of Ambient Assisted Living. Thanks to Dr. Fabio Mori, the head of the Cardiovascular Imaging Unit at the Careggi Hospital in Florence, for his precious collaboration and support.

I would also like to thank present and past people in the Software Science and Technology Laboratory (STLab) at the University of Florence: Marco Biagi, Irene Bicchierai, Andrea Bonacchi, Matteo Lapini, Tommaso Magherini, Sandro Mehic, Carlo Nocentini, Manuel Pagliai, Marco Paolieri, Stefano Pepi, Alessandro Pinzuti, Lorenzo Ridi, Francesco Scutifero, Fabio Tarani, Jacopo Torrini. Thanks to Sara Fioravanti and Simone Mattolini: it has been a pleasure to work with you. A special thank goes to Laura Carnevali, for her valuable and helpful suggestions, and to Valeriano Sandrucci, for his support and for the time spent talking about software architectures.

Thanks to my parents and my sister Giuliana: you made me what I am today with your presence in my life.

Last but not least, thanks to *Maria Grazia*, for just being... *you*!

Abstract

Over the last decade, a growing digital universe of unstructured or semi-structured human-sourced information, structured process-mediated data, and well-structured machine-generated data, encourages the adoption of innovative forms of data modeling and information processing to enable enhanced insight, decision making, and process automation applied to a variety of different contexts. Healthcare comprises a notable domain of interest, where the availability of a large amount of information can be exploited to take relevant and tangible benefits in terms of efficiency of the care process, improved outcomes and reduced health system costs. However, due to the complex nature of clinical data, a number of challenges needs to be faced, mainly related on how data characterized by volume, variety, variability, velocity, and veracity can be effectively and efficiently modeled, and how these data can be exploited for increasing the domain knowledge and supporting decision-making processes.

The aim of this dissertation is to describe the crucial role played by software architectures in order to overcome challenges posed by the healthcare context. Specifically, this dissertation addresses the development and applicability of *multi-level meta-modeling architectures* to various scenarios of eHealth, where flexibility and changeability represent primary requirements. Meta-modeling principles are concretely exploited in the implementation of an adaptable patient-centric Electronic Health Record (EHR) system to face a number of challenging requirements, including: adaptability to different specialities and organizational contexts; run-time configurability by domain experts; interoperability of heterogeneous data produced by various sources and accessed by various actors; applicability of guideline recommendations for evaluating clinical practice compliance; applicability of Activity Recognition techniques for monitoring and classifying human activities in pervasive intelligent environments.

Contents

Introduction

Over the last decade, a growing digital universe of *human-sourced information* (i.e. unstructured or semi-structured data produced by human interactions, like social networks and blogs, personal documents, pictures and videos, Internet searches, emails), *process-mediated data* (i.e. structured data produced as a result of business activities by traditional ICT systems, like business and medical records, e-commerce transactions, banking movements), and *machine-generated data* (i.e. well-structured data produced on real-time by sensors and computer systems, like sensor records, system logs, Internet of Things (IoT) data) encourages the adoption of innovative forms of data modeling and information processing to enable enhanced insight, decision making, and process automation applied to a variety of different contexts [19].

Healthcare comprises a notable domain of interest, where the availability of a large amount of unstructured, semi-structured and structured information (including untapped data collected but not yet analyzed, named *dark data*) can be exploited to take relevant and tangible benefits, including: early identification of comorbidities as well as worsening health states; improvement of operational efficiency and patient outcomes; evidence of effectiveness and safety of therapeutic strategies; reduction of rehospitalization and, consequently, health system costs; real enactment of predictive models for diagnosing, treating, and delivering care.

However, a number of challenges needs to be faced, closely related to the complex nature of data to deal with, and whose characteristics can be summarized in five “Vs” [80]:

- *Volume*: the number of concepts that comprise the whole medical ontology and the number of related clinical observations to take into account are huge. The *Systematized Nomenclature of Medicine (SNOMED)* [16] contains a computer-processable collection of more than 350 000 medical terms and over 1 million relationships, to provide definitions, codes, and synonyms which cover various aspects of the medical domain, including anatomy, diseases, findings, and procedures.

- *Variety*: clinical data are heterogeneous and come in a variety of different formats, from unstructured to semi-structured and well-structured data.

- *Variability*: the medical domain is open-ended and constantly changing due to the progressive literature review process. On the one hand, new evidences on unknown or partially-documented research areas can be discovered, leading to an in-breadth extension of the domain. On the other hand, new finer-grained evidences on already documented fields can become relevant, leading to a more in-depth extension of the knowledge.

- *Velocity*: it refers to the speed at which clinical data are produced and the rate at which they should be evaluated. Digital sensors, medical devices, and smart meters are driving the need to deal with continuous streams of data in near-real time, lashing a mounting requirement for instantaneous analytics and evidence-based arrangement [36].

- *Veracity*: the partial or complete absence of information, as well as the existence of information derived from erroneous assessments, are exhibitions of the intrinsic untrustworthiness of some sources of data. Uncertainty is frequently encountered in medical practice and needs to be faced, notably in decision-making processes [116].

Data modeling and information processing definitely represent two of the main challenges posed by the healthcare context. For this reason, they constitute the key topics of this dissertation, and the rest of this introduction is focused on describing general aspects and personal contributions in both topics under consideration.

A.1 Data modeling: challenges and personal contribution

Data modeling addresses, from a structural point-of-view, how medical concepts and clinical data can be effectively and efficiently represented within a software architecture, in order to fully exploit all the potentialities deriving from the large amount of available information.

In the common practice of software development, consolidated object-oriented systems generally represents domain concepts as distinct classes hard-coded directly into software and database models [23, 38]. This so-called *single-level* approach fits well the development of systems requiring bounded complexity of the domain ontology, rapid development, with expected low rate of change and limited evolutionary maintenance. However, it inevitably fails when system requirements and domain concepts change very often, leading to a continuous cycle of system re-coding, re-building, re-testing, and re-deploying.

Since adaptability and changeability [55] represent primary requirements in the healthcare context, a more convenient and dynamic architectural solution is needed. So-called *meta-modeling* architectures [134] can fulfill these requirements. In practice, they are designed for changing data structure and system behaviour dynamically, providing a high level of flexibility and adaptability to deal with the complex nature of data characterizing the medical domain. These architectures are usually composed by different levels of abstraction [28]: while one or more *meta levels* provide a self-representation of the system encoding knowledge about data type structures, algorithms, and relationships, the *base level* models and implements the application logic, using information provided by meta levels.

Concerning data modeling challenges, this dissertation addresses the development and applicability of multi-level meta-modeling architectures to various scenarios of eHealth, where adaptability and changeability represent primary requirements. Specifically, this work describes how meta-modeling principles are concretely exploited in the implementation of an Electronic Health Record (EHR) system [56, 47], named *Empedocle*, to allow agile tailoring for the needs of different healthcare scenarios, and to face a number of challenging requirements, including: adaptability to different specialities and organizational contexts; run-time configurability by domain experts; interoperability of heterogeneous data produced by a variety of sources and accessed by a variety of actors; usability at different levels (from health professionals to subjects-of-care). Experimentations on real outpatient and home care scenarios are reported to demonstrate feasibility and effectiveness of adaptable EHR systems applied to eHealth, with a preliminary focus on performance issues that can affect meta-modeling architectures.

A.2 Information processing: challenges and personal contribution

Information processing addresses, from a functional perspective, how all available data can be exploited for supporting advanced mechanisms of data mining, including inferring new information and knowledge from known facts and evidences, and enabling decision-making processes.

This dissertation investigates information processing in two different scenarios, i.e. healthcare and Ambient Assisted Living (AAL) contexts.

A.2.1 Decision Support Systems and Clinical Practice Guidelines

In the healthcare context, *Clinical Decision Support Systems (CDSSs)* [58] play a major role for their potential to efficiency improve the quality of medical decisions, reducing medical errors and supporting all subjects involved in the care process. Moreover, the use of CDSSs in combination with evidence-based clinical practices promises to substantially improve healthcare quality, leading to a real implementation and effective enactment of Clinical Practice Guidelines (CPGs) in the organizational workflow [46, 31, 95].

Guidelines are designed to support decision-making processes in healthcare, and clarify areas of consensus through a systematic revision of clinical evidence, so as to improve health outcomes, reduce clinical risk, as well as support clinical evaluation procedures for establish the degree of compliance of courses of action with respect to the best practices established by CPGs.

In order to provide patient-specific advices based on existing clinical data and support automated compliance evaluation of the quality of clinical processes, this dissertation addresses the development and applicability of a CPG-driven EHR system, in which recommendations extracted from medical guidelines (and clinical protocols) are integrated with clinical data collected within an EHR system. A two-step process for automated compliance evaluation of clinical processes is built on top of this architecture, and validated with respect to real medical guidelines.

A.2.2 Information processing for Activity Recognition

Due to the increasing ageing of population and the prevalence of chronic diseases, continuity and supportive care delivered at home assumes a growing relevance, providing a means to improve quality of life, to optimize costs and services, and to delay or discontinue the access to hospitalization and specialized health structures.

In this context, Ambient Assisted Living (AAL) [24] encompasses technical systems, infrastructures, and services to support elderly people in their daily routine, to allow an independent and safe lifestyle, as long as possible, via the seamless integration of information and communication technologies within homes and residences. AAL technologies are today being developed, aiming to implement safe environments around assisted peoples [101], and help them maintaining independent living. Most efforts towards the realization of AAL systems are based on developing pervasive devices and use Ambient Intelligence (AI) [36] to integrate these devices together.

Activity Recognition (AR) [67, 32] is an important area in the context of AAL. The aim of AR is recognizing common human activities starting from events generated by a variety of remote sensors deployed in a smart environment [32, 91], in order to evaluate the health status of chronically ill and ageing populations, and prevent adverse outcomes.

Regarding the topic of applying information processing techniques to the AAL context, this dissertation describes: first, how a meta-modeling architecture can be exploited to deal with the variety of different sensed data produced by a pervasive intelligent environment; secondly, how a continuous-time model-based approach for recognizing human activities can be built on top of this architecture, taking into account not only the sequencing and types of sensed events but also the continuous duration of activities and of inter-events time. The proposed approach is validated with reference to a public dataset widely used in applications of AAL [125], providing results that show comparable performance with state-of-the-art AR discrete-time approaches.

The dissertation is organized in four chapters.

- Chapter 1 revises characteristics, peculiarities and limits of multi-level meta-modeling architectures [28, 134], illustrating how adaptable systems underlying a meta-modeling approach make the difference in application domains characterized by flexibility and run-time configuration as primary requirements.

- Chapter 2 illustrates how meta-modeling architectures support design and implementation of adaptable Electronic Health Record (EHR) systems [56, 47] in order to face a number of challenging requirements posed by the healthcare context, and how automated compliance techniques can be effectively modeled into a guideline-driven EHR system, and exploited for evaluating the quality of the clinical process [40].

- Chapter 3 discusses how patient-centric Electronic Health Record (EHR) systems underlying a meta-modeling architecture can be efficiently cast into the home care scenario for supporting personalized processes of care [111, 48], and how adaptable systems can be exploited in combination with Activity Recognition (AR) techniques [91, 67, 86, 125] for monitoring and classifying human activities starting from data generated by sensors deployed in a smart environment [32].

- Chapter 4 explores meta-modeling architectures from a performance point-of-view, investigating the performance impact that new persistence approaches based on promising NoSQL technologies [115, 2, 3] can bring in the model-driven re-engineering of a meta-modeling architecture with respect to consolidated relational solutions.

Chapter 1

Meta-modeling architectures for adaptable systems

This Chapter revises characteristics, peculiarities and limits of multi-level meta-modeling architectures [28, 134], illustrating how adaptable systems underlying a meta-modeling approach make the difference in application domains characterized by flexibility and run-time configuration as primary requirements.

1.1 Motivations

Support for variation is the key to sustainable architectures for long-lived applications, mainly in domains characterized by high volatility and flexibility, where software requirements are not stable but evolve over time. Since the need for variation can occur at any time and can affect structural as well as functional aspects of a system, specifically during the production stage, it is hard to forecast future requirements and interventions and how they can impact on the overall architecture.

As a result, designing a system that meets a wide range of different requirements a priori can be an overwhelming task (e.g. changing software is tedious, error prone, and often expensive), resulting as an unrealistic solution in the practice. This *static* approach inevitably fails when new and neglected requirements emerge, leading to a continuous cycle of system changing, re-building, re-testing, and re-deploying. Therefore, it is clear that the complexity deriving from particular variations should be hidden from maintainers through a uniform mechanism able to deal with different types of changes.

For this reason, a better solution is recurring to more abstract architectures open to modification, extension and evolution over time. In the literature, this kind of adaptable architectures are called *meta-modeling* architectures or *reflective* architectures [28, 134]. Systems resulting from the exploitation of opportunities offered by meta-modeling architectures can be easily adapted to changing requirements on demand, reducing significantly the number of interventions to accommodate requested variations, and making the whole system more stable and maintainable.

The literature reports a number of real usages and examples application of adaptable systems that emphasize flexibility and run-time adaptability via a meta-modeling architectural style, covering a variety of different domains: from generic frameworks for representing and manipulating attribute composite objects [59], to health information systems for collecting clinical observations [41], to versatile e-commerce platforms for managing a wide range of business transactions [42].

1.2 Benefits and limits

Exploiting meta-modeling principles for the design and implementation of adaptable systems provides various benefits.

On the one hand, the reference model, which defines structure and semantic of system information, becomes more concise and stable, consisting of a relative small number of classes that represent only non-volatile domain concepts. The whole development cycle is speeded up, reducing significantly the number of required maintenance interventions. On the other hand, while the reference model is developed and maintained by software engineers, knowledge concept definitions are directly provided by experts of the application domain, overcoming misunderstandings between domain specialists and software developers.

However, modeling systems that emphasize changeability and adaptability [55] as primary requirements, and meeting a variety of current and future requirements inevitably result in a more complex software architecture, with various drawbacks.

First, the reference model becomes more indirect and than less intuitive (e.g. more general and abstract concepts to deal with), adding extra complexity to the design of the whole architecture [8]. In so doing, the system become harder to understand, code, test, and maintain.

Secondly, developing adaptable systems implies some relevant implementation challenges [134], that should be faced closely:

- mapping the high-level reference model into a low-level data layer, in order to make the model persistent;
- adapting Graphical User Interfaces (GUIs) to volatile domain concepts at run-time;
- supporting system maintenance for both software developers and domain experts, through the use of specific tools and GUIs.

Last but not least, a meta-modeling architecture is often exposed to major performance inefficiencies, due to the high degree of abstraction of the underlying meta-model, requiring extra processing and instantiation, at run-time, of an increased number of objects (and relationships) for describing the whole domain.

1.3 Pattern-based solutions

A pattern-oriented architectural design can partially mitigate hurdles resulting from the increased complexity induced by the application of meta-modeling principles. Positive consequences are mostly linked to the quality of code, and notably to maintainability, reusability, and consolidated understanding of implementation choices and consequences. In addition, the reuse of general solutions to commonly occurring problems in the design of adaptable systems supports and helps software developers without any appropriate skills and experiences regarding these kind of architectures.

A brief description of most notable pattern solutions for meta-modeling architectures is provided as follows.

1.3.1 The Reflection architectural pattern

The *Reflection* pattern provides a mechanism for changing structure and behavior of software systems dynamically [28], addressing from an architectural point-of-view how systems with a wide range of different requirements (usually not investigated in advance) can be efficiently built. This can be achieved via a two-layer architecture, as depicted in Fig. 1.1.

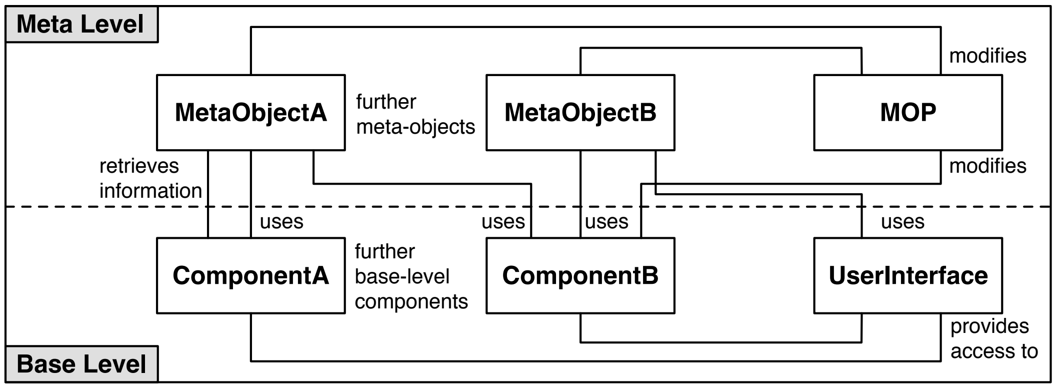


Figure 1.1. The general structure of a meta-modeling architecture as formalized by the *Reflection* architectural pattern [28].

Specifically, while a *meta level* provides a self-representation of the system encapsulating information about aspects that are likely to change over time (e.g. type structures, algorithms, function call mechanisms) into a set of so-called *meta-objects*, and making, in this way, the system self-aware, a *base level* implements the application logic using meta-objects to remain flexible (i.e. not need to hard-code information). In so doing, changes to information kept in the meta level affect subsequent base level behavior, leaving the base level totally unaware of any variation.

The meta level also implements a *Meta-Object Protocol (MOP)*, i.e. a specialized interface that can be used to dynamically configure and modify meta-objects in well-defined way. In particular, the MOP performs modifications and extensions to the meta-level code, re-compiling changed parts and linking them to the application at run-time, and providing, in this way, a powerful reflective mechanism with explicit control over system variations.

The Class-Responsibility-Collaboration (CRC) cards [129] of Fig. 1.2 summarize dynamics of interaction and collaboration between *base level*, *meta level*, and *MOP* components.

Class Base Level	
Responsibilities	**Collaborators**
▪ Implements the application logic. ▪ Uses information provided by the meta level.	▪ Meta Level

Class Meta Level	
Responsibilities	**Collaborators**
▪ Encapsulates system internals that may change. ▪ Provides an interface to facilitate modifications to the meta-level.	▪ Base Level

Class Meta-Object Protocol	
Responsibilities	**Collaborators**
▪ Offers an interface for specifying changes to the meta level. ▪ Performs specified changes.	▪ Meta Level ▪ Base Level

Figure 1.2. Responsibilities and collaborations between *Reflection* components.

Finally, note that the *Reflection* pattern enables to generalize from a *two-level* architectural style to a *multi-level* meta-modeling approach, separating meta-objects in different meta levels, and recursively introducing additional meta levels on top of those that already exist.

1.3.2 The Dynamic Object Model pattern

The *Dynamic Object Model* pattern [106], also known as *Type Square* pattern [133], is a compound pattern mainly composed by two structural patterns:

- the *Type Object* pattern [60];
- the *Property* pattern [105].

The intent of this pattern is achieving a flexible and adaptable architecture through an agile manipulation of object types and properties at run-time, including adding new types/properties, and changing existing types/properties and relationships between objects.

One simple and consolidated way to model object types is as distinct hard-coded classes. In so doing, as depicted in Fig. 1.3, adding new object types and properties to the system means manually creating new subclasses and adding new attributes to existing ones.

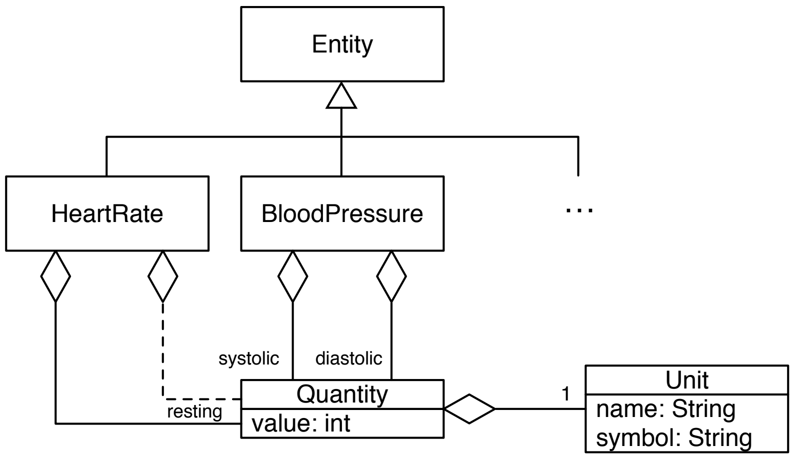


Figure 1.3. A single-level architectural solution.

However, this kind of solution (i.e. *single-level* approach) inevitably fails in domains where:

- the number of subclasses is huge and unknown at the development stage;
- the number of class attributes is not stable and changes over time.

In order to avoid intensive maintenance interventions on software code, the *Dynamic Object Model* pattern exploits the advantages offered by the combination of *Type Object* and *Property* to make the best use of both patterns.

On the one hand, the *Type Object* pattern provides a way to dynamically generate new object types (i.e. meta-objects according to the *Reflection* formalism) just representing them as instances of a generic `EntityType` class instead of creating new hard-coded classes. This pattern works very well when:

- instances of a class need to be grouped together according to their common attributes and/or behavior;
- a large number of subclasses is required and/or the variety of subclasses is unknown;
- new groupings, not predicted during the design phase, needs to be created at run-time;
- a new subclass needs to be created after its instantiation without having to mutate it to a new class;
- groupings needs to be recursively nested.

On the other hand, the *Property* pattern provides a way to dynamically generate new properties just representing them as instances of a generic `Property` class instead of adding new attributes to existing classes.

The meta-modeling architecture of Fig. 1.4 is finally obtained applying twice the *Type Object* pattern: the first time for splitting the `Entity` class from its `EntityType`, and enabling dynamic instantiation of new object types at run-time (e.g. new medical measurements like `BloodPressure` or `HeartRate`);

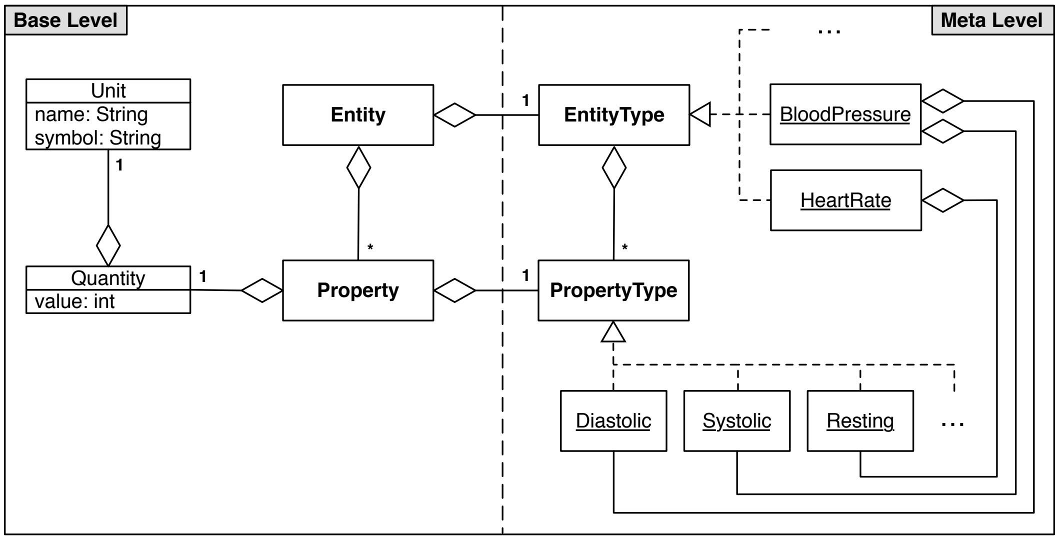


Figure 1.4. The general structure of a meta-modeling architecture as formalized by the *Dynamic Object Model* pattern [106].

the second time for splitting the `Property` class from its `PropertyType`, and enabling dynamic instantiation of new object properties at run-time (e.g. new parameters for `BloodPressure` like `Diastolic` or `Systolic`).

Summarizing, the *Dynamic Object Model* pattern offers significant benefits:

- supporting run-time creation and change of object types and properties;
- providing independent sub-classing;
- avoiding subclass explosion;
- delegating domain experts to actively contribute to maintain and keep up-to-date the system.

1.3.3 The Observations & Measurements analysis pattern

The *Observations & Measurements* pattern [41] represents a real implementation of meta-modeling architectural principles as formalized by [28] and implemented by [106, 133], applied to the clinical process. Specifically, this pattern comprises an embodiment of the *Reflection* pattern, where meta-objects are used to create abstraction on the attributes carried by different object types.

As shown in Fig. 1.5, `Measurement`, that allows to record quantitative information (e.g. heart rate, blood pressure), and `CategoryObservation`, that extends the expressiveness of the pattern for taking into account qualitative information (e.g. blood group, gender), are both represented in a so-called day-to-day *operational level*. Otherwise, their configuration, in terms of semantic definition of domain concepts (i.e. `PhenomenonTypes`), is constrained by a so-called *knowledge level*, where changes are typically more infrequent.

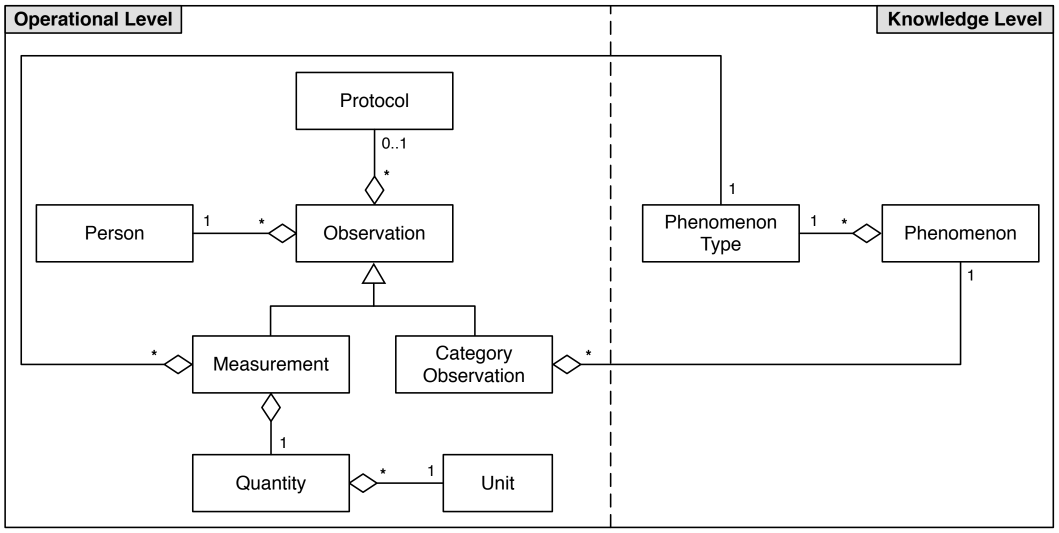


Figure 1.5. The general structure of a meta-modeling architecture as formalized by the *Observations & Measurements* analysis pattern [41].

Note that the *Observations & Measurements* pattern exploits *Dynamic Object Model* principles to separate clinical `Observations` from their `PhenomenonTypes`.

Chapter 2

Adaptable systems in healthcare

This Chapter illustrates how meta-modeling architectures support design and implementation of adaptable Electronic Health Record (EHR) systems [56, 47] in order to face a number of challenging requirements posed by the healthcare context, and how automated compliance techniques can be effectively modeled into guideline-driven EHR systems and exploited for evaluating the quality of the clinical process [40].

2.1 Exploiting adaptability for modeling heterogeneous clinical data

Adaptability and changeability [55] represent primary requirements in the healthcare context, due to the complex nature of the medical domain. The intrinsic complexities exposed by the health context are mainly derived by the essence of clinical data to deal with. In order to efficiently represent and process heterogeneous data, in which variability and variety are not-trivial characteristics, a meta-modeling architectural solution is strongly demanded, mainly for driving the design and implementation of health information systems, like Electronic Health Record (EHR) systems, which scope is collecting and processing clinical data to improve the clinical practice.

2.1.1 Electronic Health Record systems

In the systems perspective, an *Electronic Health Record (EHR) system* [56] is a software for recording, retrieving and manipulating information in terms of Electronic Health Records (EHRs), i.e. repositories of retrospective, concurrent, and prospective information regarding the health status of a subject-of-care, in computer processable form.

In an architectural point-of-view, an *EHR* system is defined as a set of components that form the mechanism by which EHRs are created, used, stored and retrieved including people, data, rules and procedures, processing and storage devices, and communication and support facilities.

The primary purpose of EHR systems is to share patient health information between authorized users for supporting continuing, efficient and quality integrated health care, facilitating communication among involved healthcare professionals, and achieving better patient outcomes.

Secondary purposes include:

- *quality of care management*, in terms of outcome assessment (e.g. hospitalization and death rate) and technical performance (e.g. competence of clinicians, compliance to the best practices established by medical guidelines and legislation);
- *medical education* and training related to the practice of being a medical practitioner;
- *research*, in terms of experimentation and evaluation of new diagnostic modalities, disease prevention measures and treatments, epidemiological studies, population health analysis, ...;

- *public and population health monitoring*, in terms of accessing to quality information for effective determination and management of real and potential public health risks.

2.1.1.1 Challenges of EHR systems

Structured vs. free-text data Clinical information can be designed within an EHR system as *structured* or *free-text* data [96].

The term *structured* data refers to data that are easily identifiable because organized in a well-defined structure. Structured data have the advantage of being easily entered, stored, and searchable by search engine algorithms, providing interesting opportunities for decision support and information retrieval systems.

On the other hand, *free-text* or *unstructured* data have no identifiable structure, but they allow to represent clinical information in a more legible and accessible way that better suits human natural language, mainly when patients are directly involved in the care process. But on the down side, unstructured data are more prone to errors and misunderstandings, not paying enough attention to machine-processable operations.

A *mixed* solution combining structured and free-text data is often a good compromise between readability offered by a free-text representation, and reusability and completeness provided by structured data, mainly when clinical information are subjected to changes over time.

Adaptability to different contexts-of-use Different types of EHR have been formally defined [56, 47], some well-described by standards and organizations, some derived from specific contexts-of-use and health scenarios in which they are largely employed. The main differences are in terms of which kind of information is managed, which level of granularity is used for representing clinical data, and which subjects-of-care are involved:

- *Electronic Medical Record (EMR)* is an EHR generally focused on medical care, further specialized in:

  - *departmental EMR*, containing patient's clinical information from a specific hospital department [66];
  - *inter-departmental EMR*, containing patient's clinical information shared between hospital departments [96];
  - *hospital EMR* or *Electronic Patient Record (EPR)* or *Computerized Patient Record (CPR)*, containing patient's clinical information from a specific hospital;
  - *inter-hospital EMR*, containing patient's clinical information shared between hospitals.

- *Personal Health Record (PHR)* contains information partly or totally entered by the patient [118];
- *Computerized Medical Record*, an electronic version of paper-based medical records, obtained by scanning original documents;
- *Population Health Record (PopHR)* is a repository of statistics, measures, and indicators regarding the state of and influences on the health of a defined population, in computer-processable form, stored and transmitted securely, and accessible by multiple authorized users [43].

Due to the variety of scenarios in which EHR systems operate, changeability and adaptability [55] are qualities of primary relevance, that largely influence the ability of a software product in fitting the needs of different specialities and organizational contexts.

Inversion of responsibility Since medical domain is extensive and subject to evolution over time, the healthcare context is characterized by high volatility in terms of medical concepts needed to be taken into account, demanding agile adaptability and run-time configuration from EHR systems.

These requirements are fulfilled empowering some qualified users in the medical domain to directly contribute their knowledge for configuring the EHR domain ontology, without requiring any programming skill. This realizes a powerful *inversion of responsibility* process, where medical experts can change system structure and behavior, improving system *maintainability* and reducing efforts and delays induced by the intermediation of ICT experts.

Foundational vs. functional vs. semantic interoperability Sharing information between different users belonging to the same organizational context or to different departments/hospitals is essential to achieve a high level of quality about health services provided to a subject-of-care. To this end, interoperability between EHR systems is mandatory, and can be achieved at three different levels of abstraction [74]:

- *foundational level* is the basic level of interoperability, in which two or more EHR systems are able to exchange clinical information, without requiring any ability by receiving systems to interpret exchanged data;
- *structural/functional level* is an intermediate level that defines the format (i.e. syntax) of data exchange, ensuring the ability by EHR systems to make effective use of structured data;
- *semantic level* is the highest level of interoperability and provides the ability for information shared by EHR systems to be interpreted at the semantic level of domain concepts, enabling receiving systems to apply automated computer processing and inference mechanisms.

Knowledge inference *Decision Support Systems (DSSs)* [98] are a specific class of computerized information systems that supports business and organizational decision-making activities. A *Clinical Decision Support System (CDSS)* is an adaptation of the DSS commonly used to support business management, helping health professionals make clinical decisions through the generation of case-specific advices (e.g. determining the nature of a patient's disease state or formulating a therapeutic plan), starting from patient clinical data [113, 12]. There are two main types of CDSS [17]:

- *knowledge-based CDSS*, in which three parts can be easily identified:

  - a *knowledge base*, that consists of a set of deterministic conditions defined starting from patient's clinical information, and represented often in the form of IF-THEN rules (e.g. IF patient's body temperature is greater than 37.5 °C, THAN alert the physician about fever), or probabilistic conditions that are IF-THEN rules with probabilistic information [73, 26];
  - a *reasoning engine*, that combines the rules from the knowledge base with the patient's data for inferring new clinical knowledge and advices;
  - a *communication system*, that allows the user to input patient's data into the system and to get output results.
- *non knowledge-based CDSS* exploits machine learning techniques to learn from past experiences and/or to recognize patterns in the clinical data [68, 72].

In the common practice of CDSS, patient's data are no longer entered by users, but they are already in electronic form and come directly from EHR systems [64], where they were originally entered by health professionals, or from other integrated systems as patient administration, laboratory, pharmacy, Radiology Information System and Picture Archiving and Communication System (RIS/PACS).

2.1.1.2 Standards for EHR systems

The development and adoption of national and international standards in the context of EHR systems is essential for sharing patient's clinical information between different health professionals and supporting interoperability between information systems involved in the care process. A description of two of the well-known standards for EHR systems (i.e. openEHR [4] and HL7 [1]) follows.

openEHR It is an open standard specification in health informatics that describes the management and storage, retrieval and exchange of health data

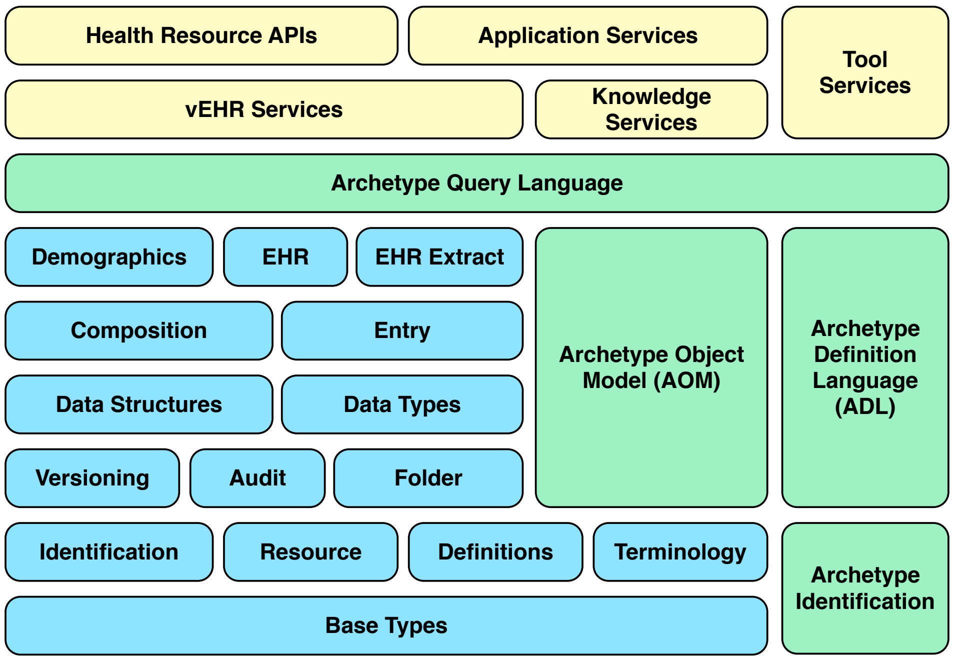


Figure 2.1. openEHR architecture overview. Blue boxes represent the Reference Model (RM), green boxes identify the Archetype Model (AM), and yellow boxes constitute the Service Model (SM).

in EHRs [4] . The abstract openEHR architecture [15] consists of the following key elements, depicted in Fig. 2.1:

- the *Reference Model (RM)* [13] is characterized by a two-level modeling approach, where a stable *Reference Information Model (RIM)* constitutes the information level, while formal definitions of clinical content in the form of *archetypes* and *templates* represent the knowledge level. This separation between knowledge and information levels allows openEHR to overcome problems caused by the ever-changing nature of clinical domain.

- the *Archetype Model (AM)* formalizes the bridge between information models and knowledge resources. In [14], Beale et al. define archetype and template concepts as follows:

  – *archetype* is "a computable expression of a clinical concept in the form of structured constraint statements, based on a reference (information) model. openEHR archetypes are based on the openEHR reference model. Archetypes are all expressed in the same formalism. In general, they are defined for wide re-use, however, they can be specialized to include local particularities. They can accommodate any number of natural languages and terminologies".

  - *template* is "a directly locally usable definition which composes archetypes into larger structures often corresponding to a screen form, document, report or message. A template may add further local constraints on the archetypes it mentions, including removing or mandating optional sections, and may define default values".

- the *Service Model (SM)* defines access to key back-end services (e.g. EHR Service and Demographics Service), while a growing set of lightweight REST-based APIs based on archetype paths are used for application access.

Health Level-7 (HL7) It refers to a set of international standards, guidelines, and methodologies for the exchange, integration, sharing, and retrieval of electronic health information that supports clinical practice and the management, delivery and evaluation of health services [1]. HL7 standards are classified in two main categories:

- *primary standards* account for system integrations and interoperability, such as:

  - *HL7's Version 2.x Messaging Standard*, where a series of electronic messages to support administrative, logistical, financial as well as clinical processes are defined. This messaging standard allows interoperability of clinical data between systems, and it is designed to support a central patient care system as well as a more distributed environment where data reside in departmental systems. It supports two forms of message encoding, i.e. a custom delimiter-based encoding and an XML encoding.

  - *HL7's Version 3 Messaging Standard*, in which a Reference Information Model (RIM) is introduced for providing an explicit representation of semantic and lexical connections that exist between information carried in the fields of HL7 messages.
  - *HL7 Version 3 Clinical Document Architecture (CDA)*, a document mark-up standard that specifies the structure and semantics of clinical documents for the purpose of exchange between healthcare providers and patients.

- *foundational standards* define tools and blocks used to build HL7 standards, such as:
  - *Reference Information Model*, an information modeling foundation for supporting the representation of all concepts in the HL7 domain of interest;
  - *Arden Syntax*, a formalism for representing and sharing procedural clinical knowledge among health professionals, information systems and institutions;
  - *HL7 Decision Support Service (DSS)*, for facilitating the implementation of clinical decision support capabilities.

In addition, for the purposes of providing functional models and specifications for the management of EHRs, some HL7 standards have been flowed into so-called *HL7 EHR profiles*.

2.1.2 The Empedocle EHR system

In order to face a number of challenges and hurdles (see Sect. 2.1.1.1) posed by the healthcare context for EHR systems, we show how meta-modeling principles can be exploited for the design and implementation of an EHR system characterized by adaptability and changeability as primary requirements [55]. The proposed EHR system, named *Empedocle*, combines the basic functionalities that comprise the expected commodity level of any EHR system, with specific requirements posed by an outpatient scenario, where a variety of medical specialities take part.

2.1.2.1 A typical outpatient scenario

The use case diagram of Fig. 2.2 summarizes tasks and responsibilities of major actors involved in a typical outpatient scenario.

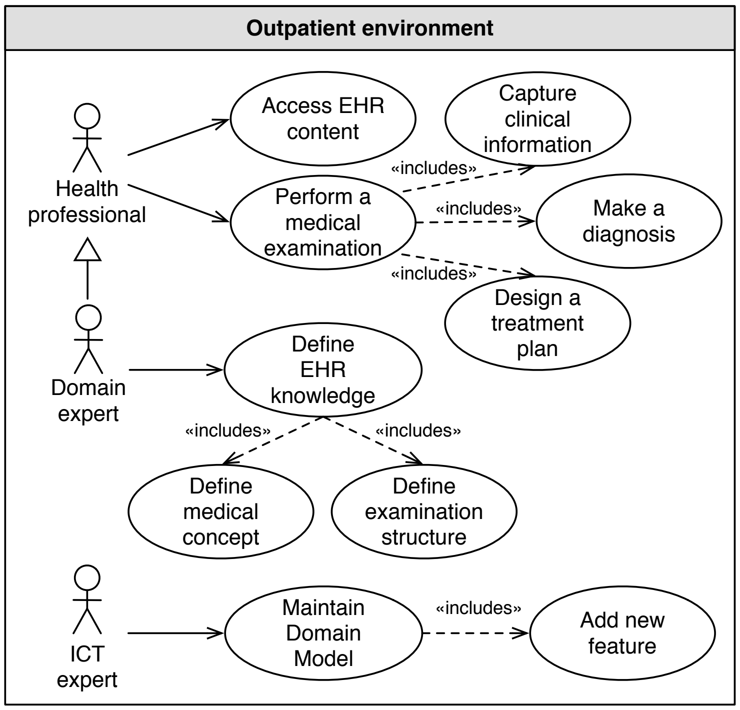


Figure 2.2. A typical outpatient scenario, specifying the major actors involved in the care process and their interaction with an EHR system.

Health professionals (e.g. general practitioner, medical specialist, registered nurse) take part to the care process at the operational level in different ways, in accordance with personal skills and specializations. Generally, a typical outpatient scenario is characterized by the following clinical steps:

1. the complete review of the patient's EHR content (e.g. clinical history, allergies, active problems, test results);

2. the acquisition of clinical data through a medical examination;

3. the formulation of a diagnosis;
4. the development of a specific treatment plan.

Medical concepts related to clinical data collected into the EHR system are identified and steadily maintained at knowledge level by one or more *domain experts*, who are health professionals with specific domain expertise as well as aware about governmental and hospital directives, and about factors depending on specialization of activities and scientific aims.

Finally, *ICT experts* play a lead role in bridging medical and informatics domains, in cases where technical skills are required for supporting health professionals through the implementation of additional system requirements that demand structural changes in the domain model, at the operational as well as the knowledge levels.

We do not report here on the characteristics of other complementary roles which are involved in the organization and enactment of the clinical process (e.g. from health direction and administrative support), but that are not directly concerned with the topic addressed in this dissertation.

2.1.2.2 A two-level meta-modeling EHR architecture

The UML class-object diagram of Fig. 2.3 provides a high-level specification of the domain model implemented in the core of the *Empedocle* EHR system. As can be seen, the architecture of *Empedocle* is based on an underlying meta-modeling scheme, where medical concepts, represented in a so-called *knowledge* level, are separated from clinical data, represented in a so-called *operational* level. These principles are obtained through a pattern-oriented design, addressed in the architectural perspective by the *Reflection* pattern [28], and in the conceptual perspective by the *Observations & Measurements* pattern [41] (see Chapter 1 for more details about pattern-based solutions).

At the *operational* level, an `EHR` represents a structured collection of health information items about a `Patient`, derived through a set of clinical `Examinations`. Specifically, during each `Examination`, a series of clinical information items like signs (i.e. objective evidences noticed), symptoms (i.e. subjective

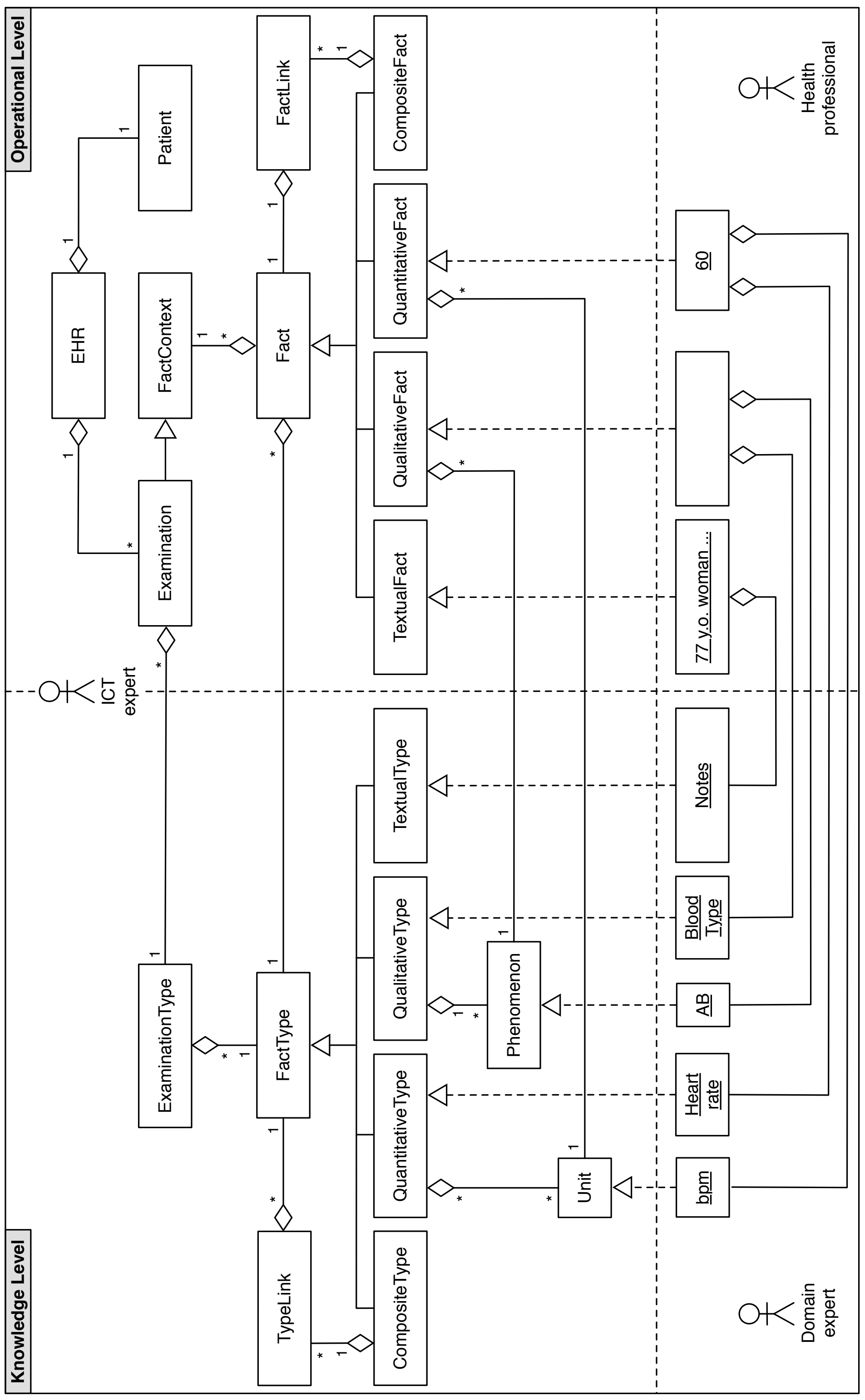


Figure 2.3. The two-level domain model of the *Empedocle* EHR system.

evidences reported by patient), and other clinical observations are captured by health professionals as instances of the `Fact` class.

Conversely, the *knowledge* level must be designed so as to accommodate the intrinsic variability of the medical domain, which depends on the evolution over time as well as on differences among medical specialities. To this end, all medical concepts can be defined directly by domain experts as instances of the `FactType` class. Four different categories of knowledge can be identified:

- `TextualType`, for free-text information (e.g. patient's *anamnesis*);
- `QualitativeType`, for values in a finite range of acceptable `Phenomena` (e.g. *blood type* with groups A, B, AB, and 0);
- `QuantitativeType`, for quantities with a specified set of acceptable `Units` (e.g. *heart rate*, measured in *beats-per-minute*);
- `CompositeType`, for composing `FactTypes` in a hierarchical structure through a *Composite* pattern [44] implementation (e.g. *vital sign* including *temperature*, *blood pressure*, *heart* and *respiratory rate*).

The same categories can be identified at the operational level, in terms of `TextualFact`, `QualitativeFact`, `QuantitativeFact`, and `CompositeFact`.

The resulting high-level model abstraction allows to separate the representation of medical knowledge (i.e. the semantic of medical phenomena) from clinical data (i.e. the value assumed by a specified medical phenomenon in a specified time for a specified patient), and empowers domain experts to contribute to this knowledge in the course of system life. In so doing, the two-level separation allows a new medical concept to be accounted just through the instantiation of a new object from the `FactType` class, avoiding the need of programming new classes or class members and without any impact on the database schema or on its records. In the same manner, new clinical observations are generated as `Fact` instances, starting from the definition given by related `FactType` instances.

In this model, `ExaminationType` class represents the structure of an `Examination` in terms of which `FactTypes` (and related `Facts`) have to be considered during a medical examination; moreover, it specifies, through `TypeLink` and `FactLink` associations, the multiplicity of occurrence of each `Fact` in order to dynamically adapt the structure to multiple contexts-of-use that require a different number of instances to be recorded.

The model supports the reuse of already defined (i.e. *named*) `FactTypes`, in order to avoid type proliferation, just referencing named types in multiple parts of the structure. Alternatively, *anonymous* `FactType` instances (i.e. `FactTypes` that do not need to be referenced by others) can be used, and the definition of their structure is included inside the parent structure. As relevant consequence, as depicted in Fig. 2.4, the `FactType` structure results in a direct acyclic graph, while the derived `Fact` structure results as a tree, usually with an increased number of nodes, due to the multiplicity attribute.

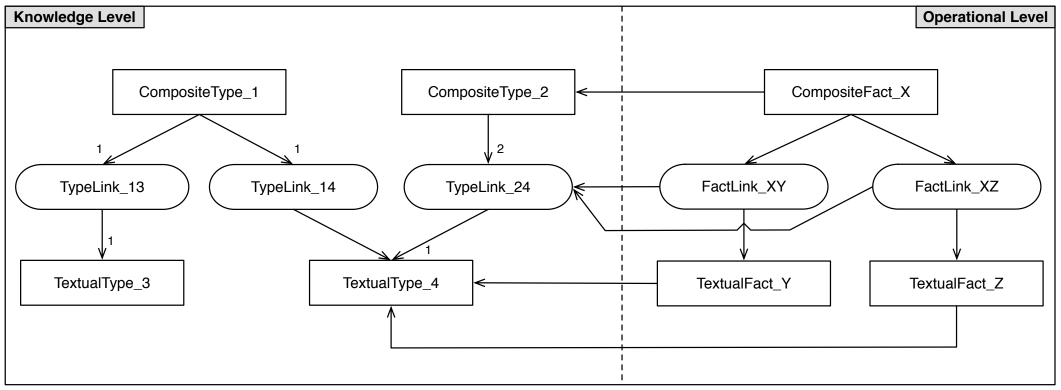


Figure 2.4. An example of `Examination` structure as represented using the domain model shown in Fig. 2.3: on the left, a direct acyclic graph obtained composing `FactTypes` and `TypeLinks`; on the right, a tree-like structure as resulting from the composition of `Facts` and `FactLinks`. Note that boxes represent instances of `FactType` and `Fact` classes and define, respectively, medical concepts and clinical observations that need to be taken into account during a clinical examination. Rounded boxes represent instances of `TypeLink` and `FactLink` classes and are used to increase the expressiveness of each `Type-to-Type` and `Fact-to-Fact` association, adding a multiplicity attribute to each relationship.

This implies a more complex data model, with various drawbacks.

On the one hand, while the number of `Facts` concretely recorded at runtime during a clinical session is bounded in semantic and multiplicity by the `FactType` definition, the real depth of an `Examination` cannot be known in advance, precluding the possibility to exploit optimized ad-hoc mechanisms for retrieving all the data, requiring instead to explore the entire structure.

On the other hand, since the model is split in two levels, the whole `Examination` will be completely known only when both parts will be provided. For this reason, retrieving all the data collected during an `Examination` is not restricted to exploring the `Fact` tree, but requires to explore the related `FactType` graph, affecting system performance.

Finally, the resulting model consists of a relative small number of classes for representing only concrete concepts; nevertheless, the high degree of abstraction is counterbalanced by the instantiation, at run-time, of an increased number of objects required for describing the actual domain. Usually, this does not represent a problem in small and static domains, but it becomes evident in terms of performance inefficiencies in domains characterized by complexity and volatility (performance issues affecting meta-modeling architectures will be described in more detail in Chapter 4).

Fig. 2.5 shows the software architecture of the *Empedocle* EHR system, as currently deployed in various outpatient clinics of the Careggi Hospital in Florence, the major hospital of Tuscany Region.

The scheme corresponds to a typical 3-tier web application, with an added *Mapping Layer* between the *Domain Layer* and the *Persistence Layer*. While the *Persistence layer* deals with data storing and persistence, and the *Domain Layer* encodes the real-world business rules that determine how data can be created, displayed, stored, and changed, the *Mapping Layer* reconciles the object-relational impedance mismatch between objects and relational data [54]. The *Presentation Layer* implements interfaces and logic for the interaction with system users through suitable GUIs.

Fig. 2.6 depicts a static view of the run-time configuration of the *Empedocle* EHR system, in terms of processing nodes and software components.

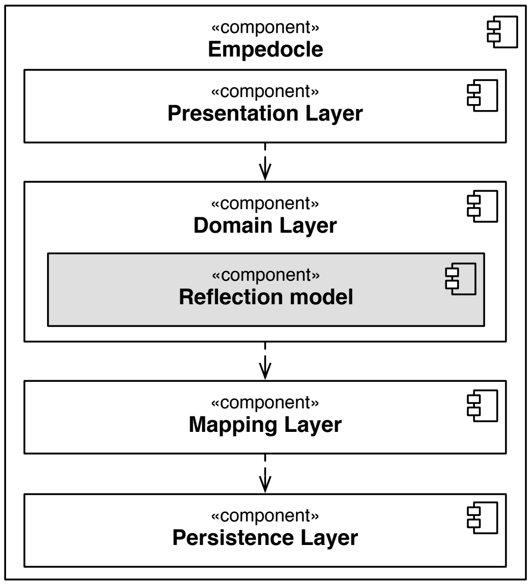


Figure 2.5. The 3-tier architecture of the *Empedocle* EHR system. The *Reflection model* component implements the two-level domain logic of Fig. 2.3.

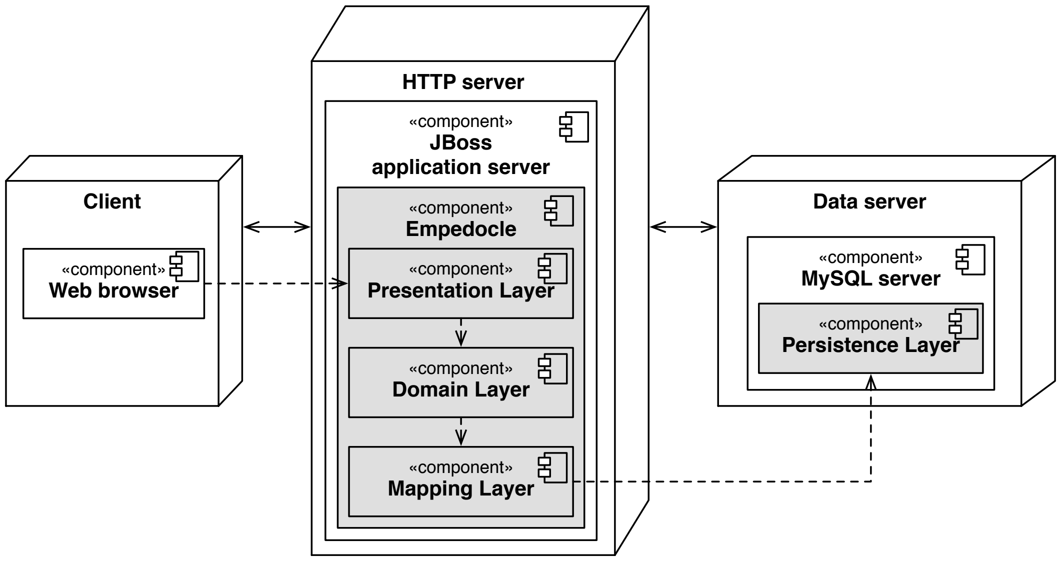


Figure 2.6. A general overview about hardware and software components and their connections for the *Empedocle* EHR system, as currently deployed at the Careggi Hospital in Florence.

2.2 Exploiting adaptability for evaluating clinical practice compliance

In the clinical practice, evaluating the quality of clinical processes is essential for improving outcomes and mitigating the clinical risk deriving from erroneous medical behaviours. Clinical Practice Guidelines (CPGs) play a fundamental role to standardize good practices and can become important reference points for supporting automated compliance evaluation of the quality of clinical processes.

2.2.1 Clinical Practice Guidelines

Clinical Practice Guidelines (CPGs) are systematically developed statements to assist practitioners and patient decisions about appropriate health care for specific circumstances [39]. Guidelines are designed to support decision-making processes in patient care, and clarify areas of consensus through a systematic revision of clinical evidence (i.e. evidence-based medicine), so as to deliver the best possible guidance to practising physicians. The intent of CPGs is to [130]:

- provide appropriate care, based on the best available scientific evidence and broad consensus;
- improve the quality of clinical decisions, reducing medical errors and clinical risk;
- improve health outcomes, reducing morbidity and mortality and increasing quality of life;
- support clinical evaluation processes, with particular attention to compliance assessment of specific courses of action with respect to clinical guidelines;
- improve consistency of care;

- empower subjects-of-care to be more informed about recommendations or treatment options;
- focus attention on key research questions and highlight gaps in the known literature;
- promote an efficient and optimized use of resources, reducing outlays for hospitalization, prescription drugs, surgery, and other procedures.

In a different way, *clinical protocols* provide more rigid criteria than CPGs, specifying how guidelines must be implemented within a specific organizational context.

In the clinical practice, real implementation and effective enactment of CPGs and local protocols may largely benefit from Decision Support Systems (DSSs) [46, 31, 95]. Since accessing information contained in conventional paper-based guidelines may be difficult, and applying it to a specific clinical case can result in a time-consuming task, a widely accepted solution is given by the creation of computer-interpretable representations of the clinical knowledge contained within CPGs, enabling guideline-based DSSs to directly operate on recommendations, and support health professionals during the decision-making process.

2.2.2 Computer-Interpretable Guideline models

In the literature, various works provide specific contributions in the formalization of models for translating CPGs in Computer-Interpretable Guidelines (CIGs). These formal models can be classified in three main categories [92]:

- *document models*, which scope is storing and organizing heterogeneous information contained in practice guideline documents into a computer-processable format (e.g. the Guideline Elements Model (GEM) [112]);
- *probabilistic models*, that apply probabilities and utility functions for analysing which decision options are most suitable for clinical cases under consideration [108, 84];

- *task-network models*, that decompose CPG algorithms into networks of component tasks that unfold over time (e.g. Asbru [78], EON [120], GLIF3 [93], GUIDE [99], PROforma [117]).

2.2.3 A guideline-driven EHR system

In order to provide patient-specific advices based on existing clinical data and support automated compliance evaluation of the quality of clinical processes, integration of guideline-based DSSs in working EHR systems is recommended. However, this implies mapping CIG concepts to EHR data with several inherent challenges [94]. Moreover, solving the trade-off problem about data completeness and system usability is equally important [89]: on the one hand, an EHR system designed to fit a restricted set of guidelines inherently induces a bias towards specific pathologies; on the other hand, representing into the EHR knowledge all medical concepts pertaining to any applicable guidelines (belonging to different specialities involved in the care process) definitely impairs the practical feasibility and usability needed by health professionals.

We report on the integration of recommendations extracted from CPGs (and clinical protocols) with clinical data collected within an EHR system, through a two-step process that jointly fulfill usability, completeness, and bias avoidance requirements [88].

In the first step, clinical information is collected within the EHR system through an unbiased general structure, adapted to a specific context-of-use (e.g. specific speciality) but not tailored to a particular guideline.

In the second step, after the formulation of a diagnostic hypothesis, CPGs applicable to the case are automatically identified, in order to evaluate the compliance of the formulated diagnosis (and the course of action leading to that diagnosis) with respect to identified guidelines. The EHR content is thus automatically adapted to focus only on medical concepts and clinical information referred into applicable CPGs. In so doing, clinical data not relevant for the diagnosis are hidden and missing information are requested.

The proposed approach is validated on top of the *Empedocle* EHR system (see Sect. 2.1.2), with reference to the *ESC/EACTS guidelines for the management of valvular heart disease* [121], providing preliminary results about feasibility and benefits.

2.2.3.1 Integrating CPGs into the Empedocle EHR system

In the proposed approach, diagnostic constraints included in CPG recommendations are manually extracted and represented within the *Empedocle* EHR system in terms of decision rules.

A *decision rule* [58] is a statement that encapsulates domain knowledge and logical flow employed in deterministic reasoning to make a decision. In a CDSS, decision rules are often represented in the form of *production rules* [57], or so-called *condition-action rules*, where the *IF-THEN* statement constitutes the conventional syntax:

$$\begin{aligned} &IF < \mathit{condition} > \\ &\quad \hookrightarrow \quad THEN < \mathit{action} > \end{aligned}$$

where $< \mathit{condition} >$ represents a logical statement that, if evaluated to true, leads to the firing of the rule, i.e. the execution of the specified $< \mathit{action} >$.

For our CPG-driven EHR system, we adopted a variation on the condition-action formalism named *Event-Condition-Action (ECA)* rule, specifying which action is triggered by the satisfaction of which condition at the occurrence of which event [85]:

$$\begin{aligned} &ON < \mathit{event} > \\ &\quad \hookrightarrow \quad IF < \mathit{condition} > \\ &\qquad\qquad \hookrightarrow \quad THEN < \mathit{action} > \end{aligned}$$

This has a number of relevant advantages:

- *understandability*: rules are represented in a way that resembles the natural language making the knowledge expressed within diagnostic constraints easier to understand by domain experts;

- *configurability*: acquisition and manipulation of CPG knowledge can be delegated to domain experts without requiring the intervention of ICT developers;

- *maintainability*: guidelines and their local adaptations need to be periodically updated due to the progressive literature review process;

- *reasoning*: the independent representation of domain knowledge and control flows stimulates the adoption of production rules by CDS systems, instead of alternative procedural solutions.

Fig. 2.7 shows how the *ECA* paradigm is conveniently cast into the *Empedocle* EHR model of Fig. 2.3.

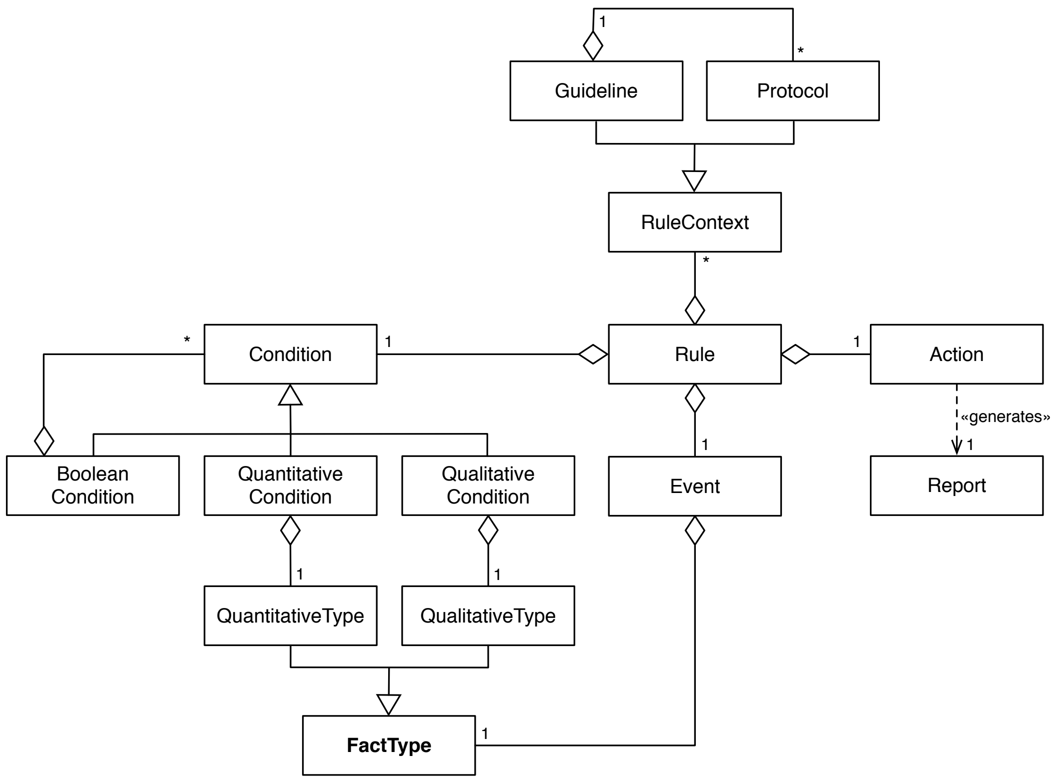


Figure 2.7. The *ECA* rule model as represented in the *Empedocle* EHR system.

In this model, a `Rule` represents an *ECA* rule in a specified `Guideline` or `Protocol` context, and it is composed by three elements:

- an `Event`, associated with a `FactType` that identifies a diagnostic hypothesis;
- a `Condition`, that specifies a logical statement starting from diagnostic constraints;
- an `Action`, performed in two steps:
    1. generation of a `Report` about outcomes achieved by the compliance process on evaluating the formulated diagnosis with respect to applicable `Guidelines`;
    2. automated adaptation of the EHR content on the basis of the formulated diagnosis.

Note that three different types of `Condition` can be expressed:

- a `QuantitativeCondition` is formulated starting from a `QuantitativeType`, and, at run-time, compares ($=, \neq, <, >, \leq, \geq$) the corresponding `QuantitativeFact` value with a fixed quantity as established by CPG recommendations (e.g. $temperature > 38\,°C$);
- a `QualitativeCondition` is formulated starting from a `QualitativeType`, and, at run-time, compares ($=, \neq$) the corresponding `QualitativeFact` value with a fixed `Phenomenon` as established by CPG recommendations (e.g. $throat = sore$);
- a `BooleanCondition` is formulated composing multiple atomic conditions through boolean operators (e.g. $temperature > 38\,°C \ \wedge \ throat = sore$).

2.2.3.2 Experimental results

Experiments were performed on the *ESC/EACTS guidelines for the management of valvular heart disease (VHD)* [121], focusing on diagnostic criteria for *aortic stenosis (AS)* and *mitral regurgitation (MR)*, which are the most frequent types of VHDs in the clinical practice [82], and more specifically, in various outpatient cardiological clinics of the Careggi Hospital in Florence, where the *Empedocle* EHR system is currently in use.

Echocardiography is the key technique used to confirm the diagnosis of VHD and to assess the degree of severity. Since VHD severity influences the prognosis and therapeutic strategy, even providing indication to surgery, we focused our investigation on *severe-AS* and *severe-MR*.

The integrative approach recommended by [121] describes a series of quantitative, semi-quantitative and qualitative echocardiography criteria to be evaluated for the diagnosis of *severe-AS* and *severe-MR*. In order to provide a specific selection of necessary and sufficient conditions for formulating a diagnostic hypothesis, an internal protocol intended as a partial implementation of [121] was formulated by the cardiological department of the Careggi Hospital. In so doing, we realized that some medical concepts involved in diagnostic constraints were not accounted by the standard cardiological examination structure as represented in the *Empedocle* EHR system, despite the high number of fields (≈ 500) composing the examination form. For these reasons, we first extended the EHR knowledge with new medical concepts required to implement the internal protocol. For the *severe-AS* case, this includes the following quantitative measurements:

- *Aortic Valve Area (AVA)*;
- *Mean Gradient (MG)*;
- *Left Ventricular Ejection Fraction (LVEF)*;
- *AVA after Low Dose Dobutamine Echocardiography ($AVA_{afterLDDE}$)*;
- *MG after Low Dose Dobutamine Echocardiography ($MG_{afterLDDE}$)*;
- *LVEF after Low Dose Dobutamine Echocardiography ($LVEF_{afterLDDE}$)*.

Than, a decision rule $R = \langle E, C, A \rangle$ for *severe-AS* is formulated as:

$$\begin{aligned} E &= \{Aortic\ stenosis = severe\} \\ C &= \{(C_1 \wedge C_2) \vee (C_1 \wedge C_3 \wedge C_4 \wedge C_5 \wedge C_6 \wedge C_7)\} \\ A &= \{report\ generation\ and\ dynamic\ adaptation \\ &\quad w.r.t.\ the\ local\ implementation\ of\ [121]\} \end{aligned}$$

where:

$$C_1 = \{AVA < 1\ cm^2\}$$
$$C_2 = \{MG > 40\ mmHg\}$$
$$C_3 = \{MG < 40\ mmHg\}$$
$$C_4 = \{LVEF < 40\%\}$$
$$C_5 = \{LVEF_{afterLDDE} > 1.2 * LVEF\}$$
$$C_6 = \{AVA_{afterLDDE} < 1\ cm^2\}$$
$$C_7 = \{MG_{afterLDDE} > 40\ mmHg\}$$

In a similar manner, more than 20 constraints for the *severe-MR* diagnosis were encoded, extending the standard cardiological examination structure for accounting missing fields.

Preliminary experimentation shows the feasibility of the proposed approach and the benefits of dynamic adaptation of the EHR content according to formulated diagnostic hypotheses, so as to support health professionals during the decision-making process, and evaluate their compliance (in terms of diagnosis and course of action leading to that diagnosis) with respect to applicable guidelines.

Ongoing activity is directed to exploit the proposed approach in the automation of compliance evaluation processes driven by retrospective analyses of large EHR databases.

Chapter 3

Adaptable systems in smart home environments

This Chapter discusses how patient-centric Electronic Health Record (EHR) systems underlying a meta-modeling architecture can be efficiently cast into the home care scenario for supporting personalized processes of care [111, 48], and how adaptable systems can be exploited in combination with Activity Recognition (AR) techniques [91, 67, 86, 125] for monitoring and classifying human activities starting from data generated by sensors deployed in a smart environment [32].

3.1 Exploiting adaptability for home care

The increasing ageing of population and the prevalence of chronic diseases push the adoption of home care processes [70] that can delay or discontinue the access to hospitalization and specialized health structures. Continuity and supportive care delivered at home assumes a growing relevance, providing a means to improve quality of life, to optimize costs and services, to reduce differences in the quality of care across wide areas far from sites of clinical excellence [103, 97]. To this end, ICT plays a crucial role through the creation of a growing ecosystem of diagnostic and monitoring devices [69, 5], communication networks, and information management applications [71], enabling novel processes of care that better fit the specific needs of patients, and effectively integrate multiple cooperating actors and multiple sources of clinical information.

Effective deployment of processes of home care and, in particular, the integration of multiple sources of clinical information emphasize the responsibility of EHR systems, which become accountable for providing effective answers to various new additional challenges [53, 118]:

- *patient-centric EHR content*, to realize advanced mechanisms of personalized medicine, and to adapt the EHR structure and behavior to the patient's health status;
- *integration of heterogeneous clinical data*, produced by a variety of sources (from humans to remote monitoring devices) and accessed by a variety of actors (e.g. healthcare professionals like general practitioners (GPs), licensed practical nurses (LPNs), physical therapists (PTs); non-healthcare professionals like patients, caregivers; support systems like human activity monitoring systems, medical alert systems);
- support for *automated compliance evaluation techniques*, in order to mitigate the clinical risk deriving from the involvement of non-healthcare professionals and also from the hurdled communication among remote subjects;
- *user-adapted EHR content*, through different yet coherent views depending on the level of competence, speciality expertise, and proximity to the patient, of each subject involved in the process;
- *interoperability of clinical data*, pertaining to different medical domains and organizational contexts.

In the software engineering perspective, most of this amounts to the achievement of a high level of adaptability and changeability [55], extending the expected commodity level of any EHR system so as to fit the specific needs posed by the home care context.

For these reasons, we faced home care challenges by adding new features to the architecture of the *Empedocle* EHR system (see Sect. 2.1.2).

3.1.1 A typical home care scenario

Fig. 3.1 summarizes the major actors partaking in the home care process.

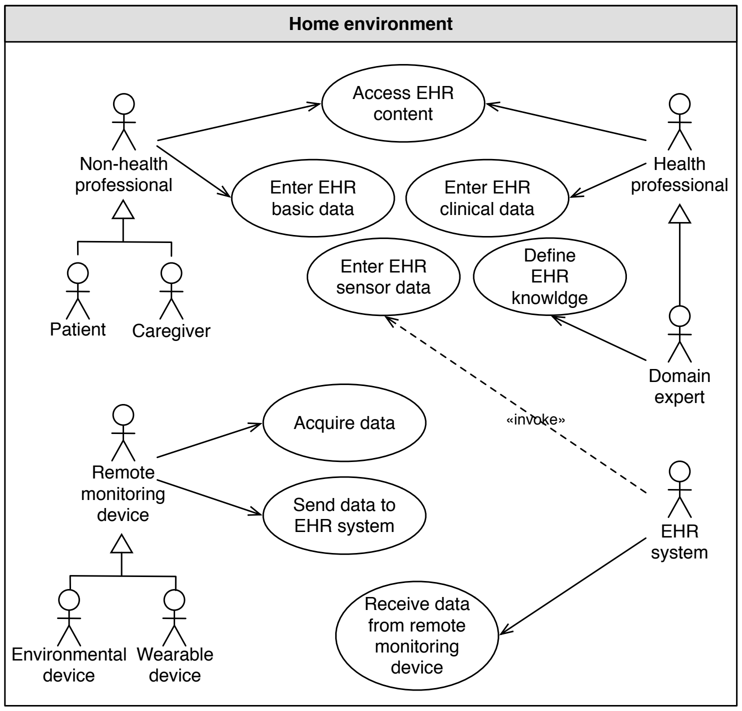


Figure 3.1. A use case diagram specifying the major actors involved in the home care process and their interaction with the EHR system.

The central role is the *patient*, for whom supportive and continuity care is provided at home. The patient can be supported during his/her Activities of Daily Living (ADLs) [65] and others activities by one or more *caregivers*.

Various *healthcare professionals* (e.g. GPs, LPNs, PTs) provide care, with the support of an integrated system of diagnostic and monitoring devices, communication networks, and information management applications. In this framework, an EHR system is responsible for the fusion of clinical information regarding the delivery of care (e.g. medical examinations, therapeutic procedures, prescriptions, treatments administration) provided by different healthcare professionals, in different fields of interest, at different times.

Remote monitoring devices for health status monitoring often comprise a primary source of information for the EHR system, specially in chronic disease management. They can be installed in the living place as *environmental devices* (e.g. for monitoring specific ADLs through the interaction of the patient with the environment), or placed directly on the patient's body as *wearable devices* (e.g. for monitoring vital signs and other health parameters like body temperature, heart rate, blood pressure), so as to realize a so-called *Body Area Network (BAN)* [61]. These devices and sensor nodes periodically send acquired data to a base station node or sink node, connected to the Internet by a home gateway, so as to feed the EHR system with these remote data.

Finally, note that the patient and other non-healthcare professionals can also access themselves the EHR content in order to consult clinical data produced by health professionals and devices, but also to report symptoms or to enter personal annotations.

3.1.2 Towards a patient-centric EHR system

The software architecture of the *Empedocle* EHR system is mainly driven by an intent of adaptability of the EHR content to clinical data characterized by changeability and volatility, that are typical requirements posed by the outpatient care context. We extended this architecture so as to fit the needs for applicability of home care practices, as depicted in Fig. 3.2.

In this model, medical concepts continue to be represented as instances of the `FactType` class, while patient's clinical data are captured as instances of the `Fact` class, acquired from different kind of `Sources`: system `Users` contribute, at the operational level, through the acquisition of clinical data

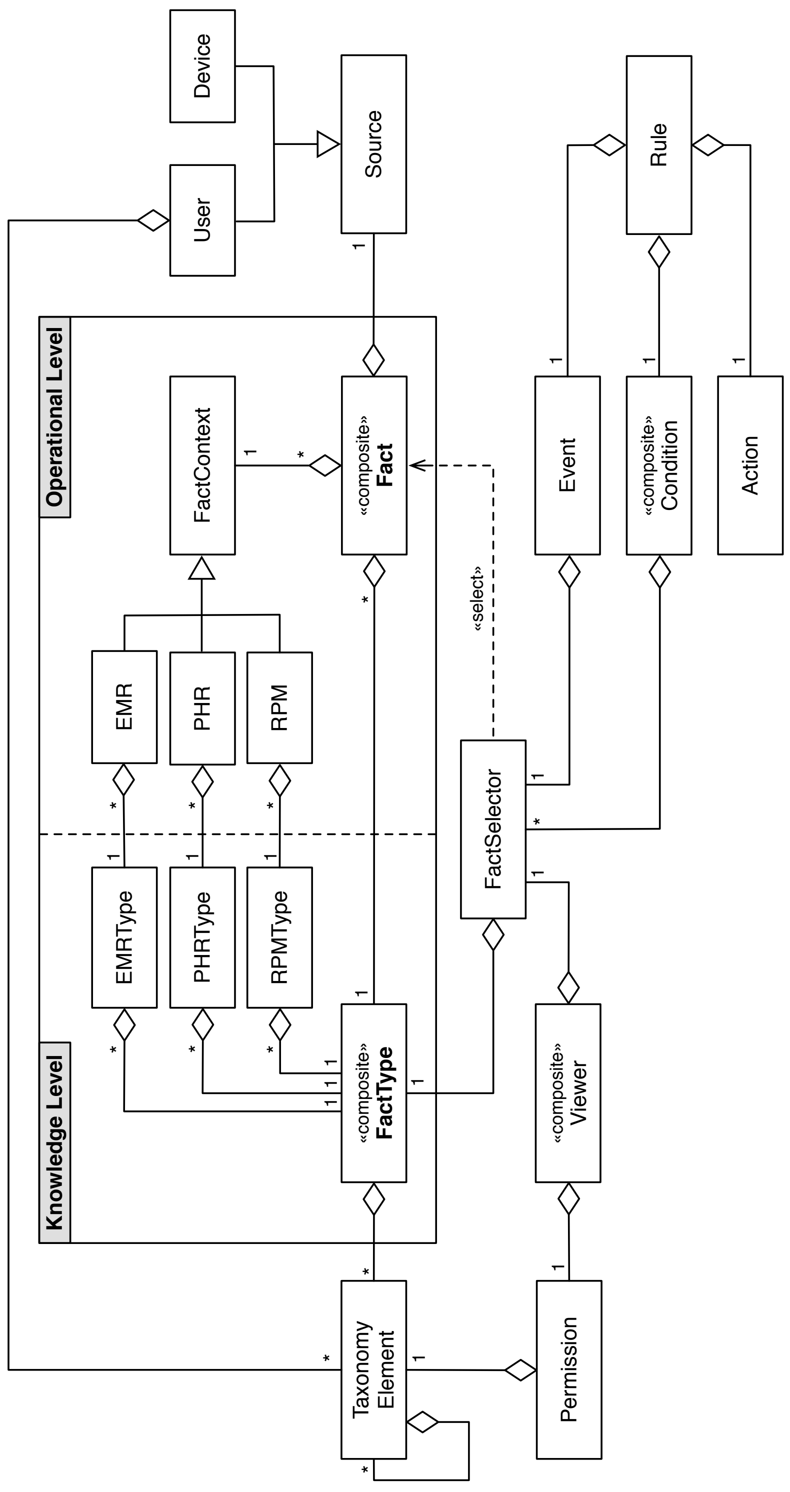


Figure 3.2. The domain model of the Empedocle EHR system adapted so as to fit the needs of the home care context.

during medical sessions carried out by health professionals, or during non-medical sessions carried out by non-health professionals; remote monitoring `Devices` feed the system with external sensed data.

Due to the variety of specialities and expertises involved in the care process, classifying medical concepts (i.e. `FactTypes`) to specific medical specialities is essential so as to support the connection and interoperability of clinical data pertaining to different medical domains. Moreover, this can be useful also for associating `Users` with organizational contexts (e.g. GP to general practice medicine, medical specialists to different medical specialization, patient and caregivers to the personal circle of the patient). To do that, a tree-like taxonomic classification, obtained through the composition of instances of the `TaxonomyElement` class, is used for tagging `FactTypes` and `Users` together. Note that, in so doing, some `FactTypes` may be tagged with multiple `TaxonomyElements`, and this means that some `Facts` may be declared to be relevant for more than one medical speciality (e.g. the observation of a *left ventricular dysfunction* can be relevant in the *Cardiology* context as well as in the *Oncology* case for starting or continuing a chemotherapy treatment) or for more than one category of `Users` (see Fig. 3.3).

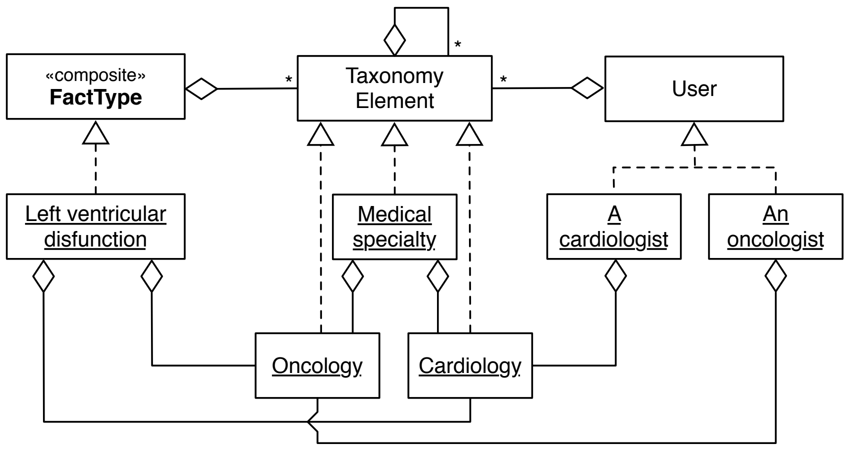


Figure 3.3. A taxonomic classification about medical specialities used for connecting medical concepts with users belonging to different specialities.

Access privileges to clinical data are established by `Permissions` that enable `Users` for all `Facts` that are tagged by some `TaxonomyElements`: in other words, only `Facts` for which a user has permissions are accessed. In addition, this allows to personalize the way in which clinical data are shown through the definition of customizable `Viewers` for each category of system `Users`, exploiting a focused selection (i.e. `FactSelector`) for identifying which `Facts` in which `FactTypes` are to be shown.

Finally, the model allows the definition of decision `Rules` based on the Event-Condition-Action paradigm [90] (see Sect. 2.2.3.1 for more details), that enables mechanisms of clinical decision support (e.g. alert messages, suggestions, deduction of new information, compliance verification) applied directly on the EHR content.

Fig. 3.4 shows the software architecture of the *Empedocle* EHR system adapted to the home care context. The 3-tier scheme is preserved, as previously depicted in Fig. 2.5 referring to the scenario of an EHR system applied to an outpatient context. Differently from the previous scenario, the *Domain Layer* now implements the domain model of Fig. 3.2, and includes a *Rule Engine* for supporting the execution of decision rules applied directly to the *Reflection model*: it accepts `Rules` and `Facts` as input, checks which `Conditions` are satisfied at the occurrence of which `Events`, and produces `Actions` as output. On top of this architecture, the *Presentation Layer* implements interfaces and logic for the interaction with users, and includes a *Viewer Engine* for automated adaptation of the EHR content to different category of users. Finally, note that *Presentation Layer* is not the only way to access the system, but is essential for human interaction. The other way to interact with the system is through an *Application Interface* that enables the system to receive data from external sources (e.g. remote monitoring devices).

3.1.3 Challenges and strategies

Personalized EHR content In principle, the meta-modeling architecture of Fig. 3.2 makes viable a full tailoring of the EHR structure to specific primary pathologies and comorbidities of each patient, up to realize advanced mechanisms of personalized medicine, and to adapt the structure on the basis of pa-

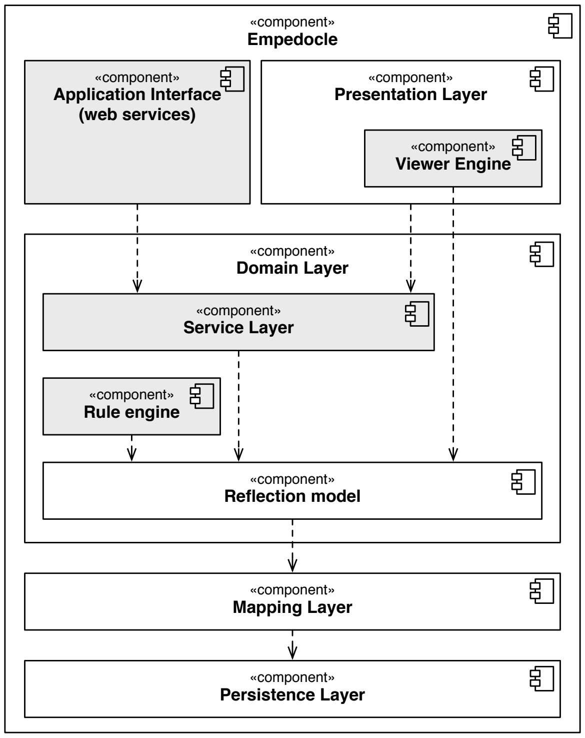


Figure 3.4. The 3-tier architecture of the *Empedocle* EHR system, extended so as to fulfill home care requirements. The *Reflection model* component implements the home care domain model of Fig. 3.2.

tient's health status. In practice, by relying on a meta-modeling architecture, all these adaptations are implemented as configurations performed through a Graphical User Interface (GUI), supporting:

- the definition of medical concepts that have to be taken into account on the basis of a specific patient's clinical condition;
- the definition of user roles and types involved in the care process;
- the definition of patient-specific decision rules;
- the definition of user-adapted views providing different EHR contents to different users.

The personalization of the EHR content is done through actions that do not require programming skills, but that can be easily delegated to experts in the medical domain. The ability to delegate to medical experts changes on the system structure and behavior comprises a major property of *maintainability* (requirement especially relevant in the home care context, in which a variety of subjects are involved at the same time), facilitating dissemination and personalization for the specific subject-of-care and the specific organizational context.

In addition, the capability of an EHR system to combine together structured and free-text fields for representing clinical data provides a mean to set the necessary trade-off between *accuracy*, *consistency* and *automated processing* provided by a structured representation and needed by health professionals, and *legibility* and *accessibility* provided by a semi-structured or completely unstructured narrative representation more close to non-professional subjects (i.e. patient and caregivers).

However, in a more practical and centralized organization, a selected set of configurations for specific chronic diseases and related medical specialities is more advisable, where predefined structures are built by domain experts with the support of ICT facilitators, and then activated and composed directly by health professionals on the basis of patient's clinical conditions.

Integration of heterogeneous clinical data The organizational context in which the problem of home care is defined gives emphasis to the integration of clinical data produced by a variety of sources and accessed by a variety of subjects. While one of the main functional requirements of an EHR system is collecting and maintaining EHR data, interoperability with remote monitoring devices represents a new challenge for this kind of systems.

The main barrier is represented by the fact that, usually, devices vary not only in relation to the observed phenomenon or measurement, but also in relation to the model manufacturer, with differences in data format and interface protocols (due to specific legacy constraints).

To overcome this problem of data adaptation, an EHR system needs the ability to be easily integrated with different technological solutions, without requiring the creation of unified bases, leveraging instead on adequate enterprise integration schemes [49]. Fig. 3.5 illustrates these concepts through the exploitation of an interfacing middleware used for implementing the transmission of sensed data from remote monitoring devices to the EHR system, so as to achieve loose coupling between software components and services. The *Event-Driven Bus* acts as an *integration broker* and mediator, able to:

1. receive events from devices, represented as *event emitters*;
2. apply format data transformations in order to mapping device legacy format to the EHR content structure;
3. marshal transformed events in requests to an *Application Interface* exposed by the EHR system (represented as *event consumer*) for receiving sensed data.

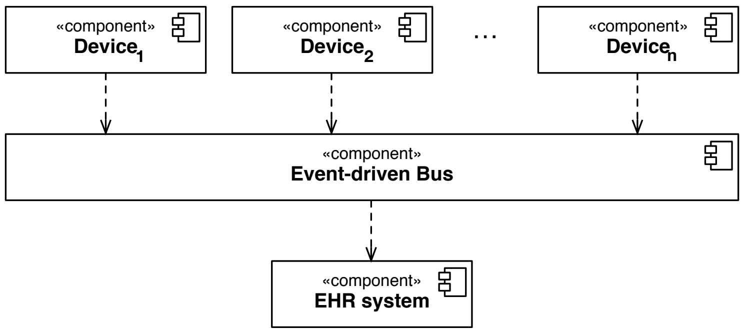


Figure 3.5. An Event-Driven Bus as integration middleware and mediator for interoperability between EHR system and remote monitoring devices.

Automated support for compliance evaluation The context of home care emphasizes the importance of control functions designed to mitigate clinical risk, aggravated in various aspects including the adoption of self-medication practices implemented directly by the patient him/herself, the involvement of non-professionals as participants in the care process, as well as hurdles induced by communication and interaction across distance. This increases the relevance of features for automatic detection of not compliant conditions in the courses of action applied to the patient with respect to expected protocols or guidelines, and for supporting efficacy and efficiency of the assistance quality control.

The integration of Clinical Practice Guidelines (CPGs) in the home care process, and their representation as decision rules within an EHR system, as described in Sect. 2.2.3.1, enables automated mechanisms of compliance evaluation of courses of action performed by a variety of actors involved in the care process, from health professionals to non-professional subjects.

User-adapted EHR content presentation As has already been mentioned, the home care model implies the involvement of a variety of subjects, including the patient him/herself, for which professional competences are not requested. This emphasizes the importance of system *usability* requirements, as a major means for the reduction of clinical risk.

An EHR system requires a user-adapted content presentation for each of the different roles involved, showing only the content for which the logged user has access privileges. In so doing, the patient and non-professional caregivers are introduced to the EHR system through the support of more personalized and simplified views, focusing the attention only on content of their interest, and reducing the risk of difficulty on understanding. Moreover, a further improvement in usability can be achieved through refinements and simplification of the look-and-feel of the user interface, often exposed to non-professional or occasional users. From an architectural perspective, this adaptation should be largely facilitated by exploiting the *Model-View-Controller (MVC)* architectural pattern [28], which allows to separate internal representations of data from the way in which data is presented to the users.

Interoperability of different medical domains The domain model of an EHR system for home care needs a unified ground for representing medical concepts pertaining to different specialities and provided by actors with different levels of expertise. This unified ground is achieved through a *shared information model* that enables and encourages communication between all subjects involved in the care process.

As depicted in Fig. 3.2, the tagging mechanism provided by `TaxonomyElements` facilitates the representation of knowledge about relevant connections among different medical concepts (i.e. `FactTypes`) and `Users` with different roles and qualifications. Moreover, `Rules` and their composing `Events`, `Conditions` and `Actions`, provide a means to implement triggering conditions on shared clinical data that can be used to execute some shared actions (i.e. raising warnings to all involved users or suggesting possible connections between users interested in the same topics, like the evolution of the patient's health status, or how a specific clinical observation varies over time).

3.2 Exploiting adaptability for Activity Recognition

Ambient Assisted Living (AAL) is becoming a well-recognized domain to help address the ageing of the population. AAL relates to the use of ICT technologies and services in both daily living and working environments aiming to help inhabitants by preventing health conditions and improving wellness, in addition to assisting with daily activities, promotion of staying active, remaining socially connected and of living independently.

Activity Recognition (AR) [67, 32] is an important area in the context of AAL. The goal of AR is to recognize common human activities, or so-called Activities of Daily Living (ADLs) [65], in real life settings, with the primary aim of evaluating the status of chronically ill and ageing populations, in relation to monitoring effectiveness of treatment over time, tracking dynamics of disabilities and preventing adverse outcomes. Accurate AR is challenging because human activities are complex (e.g. sequential vs. interleaved vs. concurrent activities; single person vs. multiple subjects) and highly different (e.g. toileting, eating, dressing).

The problem of inferring high-level activities starting from low-level sensed data generated by remote monitoring devices deployed in a pervasive intelligent environment has been investigated in many studies [32, 62, 86]. Different approaches that have been explored can be roughly categorized in data-driven, knowledge-driven and merged methodologies.

In the *data-driven* categorization, the most popular techniques which have been considered are classification models based on probabilistic reasoning (e.g. Naive Bayes (NB) classifiers, Dynamic Bayesian Networks [91], Hidden Markov Models (HMMs) [125], and Conditional Random Fields (CRF) [125]). Stochastic approaches have the advantage to be able to deal with uncertainty of sensor data, but they require a large number of training examples to provide good levels of generalization, and they need some assumptions that can limit the capability to recognize complex activities.

In the *knowledge-driven* categorization, logical modeling [33] and evidential theory [77] are the most popular techniques. They do not require a large amount of data for training because incorporate knowledge, and can handle noise and incomplete sensor data, but it may be difficult to define specifications of activities by applying only expert knowledge.

In the *merged* categorization, multiple classifiers are combined in a classifier ensemble, and individual decisions are merged in some manner to provide, as an output, a consensus decision [62].

A large part of techniques applied for AR [132] rely on or compare with Hidden Markov Models (HMM) [100], where:

- the current (hidden) activity is the state of a Discrete Time Markov Chain (DTMC), and the observed event depends only on the current activity, as depicted in Fig. 3.6;
- stochastic parameters of the model can be determined through supervised learning based on some given statistics;
- efficient algorithms are finally available to determine which *path* along hidden activities may have produced with maximum likelihood a given *trace* of observed events.

While the ground truth is often based on datasets annotated with respect to predefined activities [125, 101], more data driven and unsupervised approaches have been advocated where activities are identified through the clustering of emergent recurring patterns [102, 11]. Various extensions were proposed to encode memory in HMM by representing sojourn times through discrete general

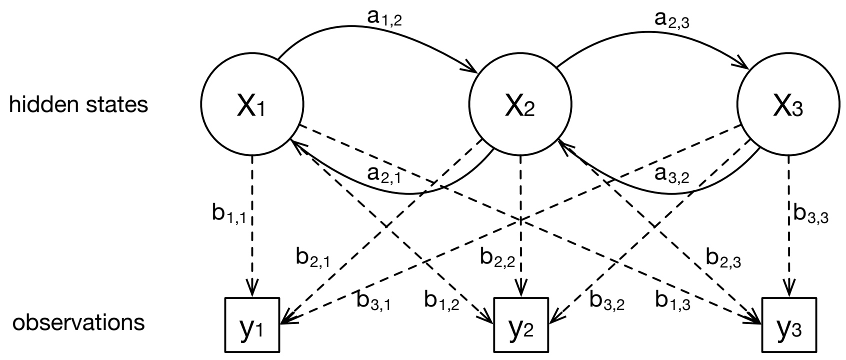


Figure 3.6. An example of Hidden Markov Model. The random variable $X(t) \in \{X_1, X_2, X_3\}$ represents the hidden state at time t. The random variable $y(t) \in \{y_1, y_2, y_3\}$ represents the observation at time t. The arrows in the diagram denote conditional dependencies: $a_{i,j}$ are the state transition probabilities, while $b_{i,j}$ represent the output probabilities.

or phase type distributions [79]. However, also in these cases, the discrete-time abstraction of the model prevents exploitation of continuous time observed between event occurrences.

To overcome this limitation, [27] proposes that the evolution of the hidden state be modeled as a non-Markovian stochastic Petri Net emitting randomized observable events at the firing of transitions. Approximate transient probabilities, derived through discretization of the state space, are then used as a measure of likelihood to infer the current hidden state from observed events. To avoid the complexity of age memory accumulated across subsequent states, the approach assumes that some observable event is emitted at every change of the hidden state. Moreover, the structure of the model and the distribution of transition durations are assumed to be given.

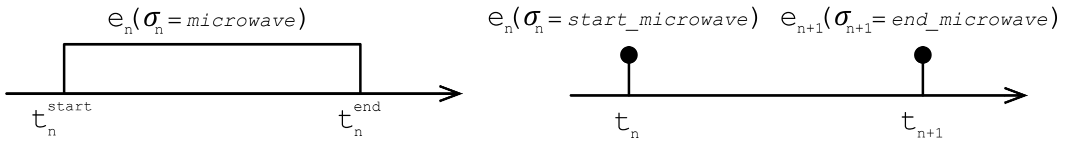
Automated construction of an unknown model that can accept sequences of observed events is formulated in [122] under the term of *process elicitation*, and solved by various algorithms [124] supporting the identification of an (untimed) Petri Net model. Good results are reported in the reconstruction of administrative workflows [123], while applicability appears to be more difficult for less structured workflows, such as healthcare pathways [75]. As a part of the process mining agenda, *process enhancement* techniques have been proposed to enrich an untimed model with stochastic parameters derived from the statistics of observed data [107].

In this dissertation, we propose a continuous-time model-based approach to online classification of ADLs that takes into account not only the type of events but also the continuous duration of activities and of inter-events time [29]. Given a stream of time-stamped and typed events, transient probabilities of a continuous-time stochastic model are used to derive a measure of likelihood for online classification of performed activities. Transient analysis based on transient stochastic state classes [51] maintains the continuous-time abstraction and keeps the complexity insensitive to the actual time between subsequent events. While the structure of the model is predefined, its actual topology and stochastic parameters are automatically derived from the statistics of observed events. Applicability to the context of AAL is validated by experimenting on a reference annotated dataset [125], and results show comparable performance with respect to offline classification based on Hidden Markov Models (HMM) and Conditional Random Fields (CRF).

3.2.1 A reference dataset for AAL

We base our experimentation on a well-known and publicly available annotated dataset for AR [125] containing binary data generated by 14 state-change sensors installed in a 3-room apartment, deployed at different locations (e.g., kitchen, bathroom, bedroom) and placed on various objects (e.g., household appliances, cupboards, doors). Seven activity types derived from the Katz Activities of Daily Living (ADL) index [65], i.e., $\Gamma = \{$*Leaving house*, *Preparing a beverage*, *Preparing breakfast*, *Preparing dinner*, *Sleeping*, *Taking shower*, *Toileting*$\}$, were performed and annotated by a 26-year-old subject during a period of 28 days. The annotation process yielded a *ground truth* consisting of a stream of activities a_1, a_2, $\dots$, a_K, each being a triple $a_k = \langle \gamma_k, t_k^{start}, t_k^{end} \rangle$ where $\gamma_k \in \Gamma$ is the activity type, t_k^{start} is the activity start time, and t_k^{end} is the activity end time. An additional activity (not directly annotated) named *Idling* is considered, consisting of the time during which the subject is not performing any tagged activity. The dataset includes 245 activity instances, plus 219 occurrences of *Idling*. Activities are usually annotated in a *mutually exclusive* way (i.e., one activity at a time), with the only exception of some instances of *Toileting* which was annotated so as to be performed concurrently with *Sleeping* (21 times) or with *Preparing dinner* (1 time).

The dataset includes 1319 sensor events, classified in 14 event types, and encoded in the so called *raw* representation, which holds a high signal in the interval during which the condition detected by a sensor is true, and low otherwise (see Fig. 3.7-left). In this case, each event is a triple $e_n = \langle \sigma_n, t_n^{start}, t_n^{end} \rangle$ where $\sigma_n \in \Sigma^{raw}$ is the event type, t_n^{start} is the event start time, and t_n^{end} is the event end time. As suggested in [10] for the handling of datasets with frequent object interaction, raw events were converted into a *dual change-point* representation, which emits a punctual signal when the condition goes true and when it goes back false (see Fig. 3.7-right). In this encoding, observations are a stream of punctual events $e_1, e_2, \ldots, e_N$ (doubled in number with respect to the raw representation, and sub-typed as *start_* and *end_*), each represented as a pair $e_n = \langle \sigma_n, t_n \rangle$, where $\sigma_n \in \Sigma$ is the event type, and t_n is the event occurrence time. In so doing, the number of events and event types has doubled, i.e., $N = 2638$, and $|\Sigma| = 28$.



$e_n(\sigma_n = microwave)$
t_n^{start}
t_n^{end}
$e_n(\sigma_n = start\_microwave)$
$e_{n+1}(\sigma_{n+1} = end\_microwave)$
t_n
t_{n+1}

Figure 3.7. Sensor representation: raw (left) and dual change-point (right).

Also for the limited accuracy of the tagging process (in [125], annotation was performed on-the-fly by tagging the start time t_k^{start} and the end time t_k^{end} of each performed activity a_k using a bluetooth headset combined with speech recognition software), the starting and ending points of activities are often delayed and anticipated, respectively. As a result, as shown in Fig. 3.8, the start (end) time of an activity does not necessarily coincide with the occurrence time of its first (last) event.

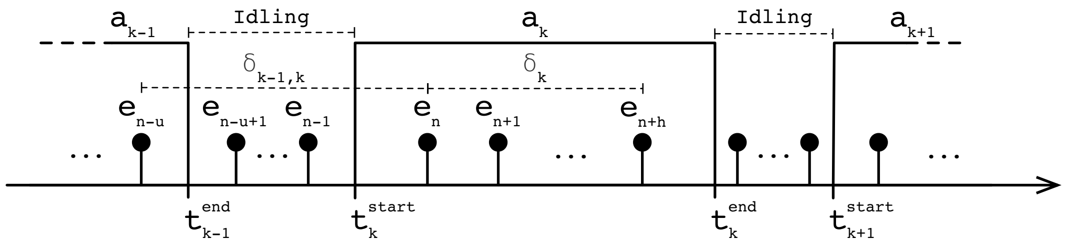


Figure 3.8. A fragment of an events stream together with annotated activities.

3.2.2 Statistical abstraction

We abstracted the dataset content so as to capture four aspects of its statistics [87]: the duration of each activity; the time elapsed between subsequent events within each activity; the type of events occurred in each activity; and, the type of events occurred as first events in each activity.

Let $\{e_n = \langle \sigma_n, t_n \rangle\}_{n=1}^{N}$ and $\{a_k = \langle \gamma_k, t_k^{start}, t_k^{end} \rangle\}_{k=1}^{K}$ be the streams of observed events and tagged activities, respectively. The duration δ_k of an activity instance a_k is computed as the time elapsed from the first to the last event observed during a_k, i.e., $\delta_k = \max_{n|t_k^{start} \le t_n \le t_k^{end}}\{t_n\} - \min_{n|t_k^{start} \le t_n \le t_k^{end}}\{t_n\}$ (e.g., in Fig. 3.8, $\delta_k = t_{n+h} - t_n$). The duration $\delta_{k-1,k}$ of an instance of *Idling* enclosed within two activities a_{k-1} and a_k is derived as the time elapsed from the last event observed during a_{k-1} to the first event observed during a_k, i.e., $\delta_{k-1,k} = \min_{n|t_k^{start} \le t_n \le t_k^{end}}\{t_n\} - \max_{n|t_{k-1}^{start} \le t_n \le t_{k-1}^{end}}\{t_n\}$ (e.g., in Fig. 3.8, $\delta_{k-1,k} = t_n - t_{n-u}$). The *duration statistic* provides mean value μ and Coefficient of Variation (CV) of the duration of each activity type, as shown in Table 3.1.

The *inter-events time statistic* evaluates the time between consecutive events occurring within an activity. In so doing, we do not distinguish between event types, and we only consider times between events. The inter-events time of *Idling* is computed taking into account *orphan events*, i.e., events not belonging to any tagged activity (e.g., e_{n-1} in Fig. 3.8), and the first event of each activity (e.g., e_n in Fig. 3.8). Also the inter-events time statistic provides μ and CV for each activity type, as shown in Table 3.1. Most of measured time spans have a CV higher than 1, thus exhibiting a hyper-exponential trend,

	Duration		**Inter-events time**	
	μ (s)	CV	μ (s)	CV
Idling	1793.102	2.039	3906.167	3.292
Leaving house	40261.455	1.042	9354.190	2.810
Preparing a beverage	35.667	1.361	7.643	2.613
Preparing breakfast	108.684	0.713	9.928	1.844
Preparing dinner	1801.889	0.640	77.966	2.589
Sleeping	26116.571	0.442	1871.836	3.090
Taking shower	485.910	0.298	102.788	1.969
Toileting	88.742	1.175	14.814	2.449

Table 3.1. Activity duration and inter-events time statistics (trained on all days but the first one).

as expected in ADL where timings may follow different patterns from time to time. Only the duration of *Leaving house* has a CV nearly equal to 1.

Table 3.2 shows the *event type statistic*, which computes the frequency $\psi_{\sigma,\gamma}$ of an event of type σ within an activity of type γ, $\forall\ \sigma \in \Sigma$, $\forall\ \gamma \in \Gamma$, i.e., $\psi_{\sigma,\gamma} = Prob\{$the type of the next event is $\sigma \mid$ the type of the current activity is $\gamma\}$.

Table 3.3 shows the *starting event type statistic*, which evaluates:

i) the frequency θ_σ of an event type σ either as the first event of an activity (regardless of the activity type) or as an orphan event, $\forall\ \sigma \in \Sigma$, i.e., $\theta_\sigma = Prob\{$the type of an event e is $\sigma \mid e$ is either the first event observed during the current activity or an orphan event$\}$;

ii) the frequency $\phi_{\sigma,\gamma}$ of an event of type σ as the first event of an activity of type γ, $\forall\ \sigma \in \Sigma$, $\forall\ \gamma \in \Gamma$, i.e., $\phi_{\sigma,\gamma} = Prob\{$an activity of type γ is started $\mid$ an event of type σ is observed and the subject was idling before the observation$\}$.

As a by-product, $\forall\ \sigma \in \Sigma$, the statistic also computes $Prob\{$the subject remains idling $\mid$ an event of type σ is observed and the subject was idling before the observation$\} = 1 - \sum_{\gamma\in\Gamma} \phi_{\sigma,\gamma}$.

	Leaving house	**Preparing a beverage**	**Preparing breakfast**	**Preparing dinner**	**Sleeping**	**Taking shower**	**Toileting**
start_front door	0.497	-	-	-	-	-	-
end_front door	0.503	-	-	-	-	-	-
start_hall-bathroom door	-	-	-	0.018	0.111	0.008	0.261
end_hall-bathroom door	-	-	-	0.023	0.115	-	0.261
start_hall-bedroom door	-	-	-	-	0.274	-	0.019
end_hall-bedroom door	-	-	-	-	0.280	-	0.013
start_hall-toilet door	-	-	0.004	-	0.041	0.540	0.057
end_hall-toilet door	-	-	0.004	-	0.045	0.452	0.063
start_cups cupboard	-	0.176	0.009	0.018	-	-	-
end_cups cupboard	-	0.176	0.009	0.018	-	-	-
start_groceries cupboard	-	-	0.119	0.055	-	-	-
end_groceries cupboard	-	-	0.123	0.055	-	-	-
start_pans cupboard	-	-	0.009	0.115	-	-	-
end_pans cupboard	-	-	0.009	0.111	-	-	-
start_plates cupboard	-	-	0.106	0.083	-	-	-
end_plates cupboard	-	-	0.106	0.083	-	-	-
start_dishwasher	-	0.010	0.004	0.005	-	-	-
end_dishwasher	-	0.010	0.004	0.005	-	-	-
start_freezer	-	0.020	0.049	0.070	-	-	-
end_freezer	-	0.020	0.049	0.070	-	-	-
start_fridge	-	0.294	0.167	0.106	-	-	-
end_fridge	-	0.294	0.167	0.106	-	-	-
start_microwave	-	-	0.031	0.023	-	-	-
end_microwave	-	-	0.031	0.018	-	-	-
start_toilet flush	-	-	-	0.009	0.067	-	0.163
end_toilet flush	-	-	-	0.009	0.067	-	0.163

Table 3.2. Event type statistic (trained on all days but the first one): frequency $\psi_{\sigma,\gamma}$ of each event type $\sigma \in \Sigma$ (rows) within each activity type $\gamma \in \Gamma$ (columns).

	Frequency θ_σ	Leaving house	Preparing a beverage	Preparing breakfast	Preparing dinner	Sleeping	Taking shower	Toileting	Idling
start_front door	0.076	0.780	-	-	-	-	-	-	0.220
end_front door	0.017	0.111	-	-	-	-	-	-	0.889
start_hall-bathroom door	0.183	-	-	-	-	-	-	0.707	0.293
end_hall-bathroom door	0.063	-	-	-	-	-	-	0.118	0.882
start_hall-bedroom door	0.050	-	-	-	-	0.148	-	0.296	0.556
end_hall-bedroom door	0.046	-	-	-	-	0.360	-	0.080	0.560
start_hall-toilet door	0.117	-	-	-	-	0.095	0.222	0.016	0.667
end_hall-toilet door	0.128	-	-	0.015	-	0.029	0.116	0.145	0.695
start_cups cupboard	0.048	-	0.308	-	-	-	-	-	0.692
start_groceries cupboard	0.044	-	-	-	0.125	-	-	-	0.875
start_pans cupboard	0.031	-	-	0.118	-	-	-	-	0.882
start_plates cupboard	0.046	-	-	0.520	-	-	-	-	0.480
start_dishwasher	0.024	-	-	-	0.077	-	-	-	0.923
start_freezer	0.020	-	0.091	-	0.273	-	-	-	0.636
start_fridge	0.068	-	0.243	0.081	0.054	-	-	-	0.622
start_toilet flush	0.039	-	-	-	-	-	-	0.524	0.476

Table 3.3. Starting event type statistic (trained on all days but the first one): for each event type $\sigma \in \Sigma$, *i)* frequency θ_σ (first column), *ii)* frequency $\phi_{\sigma,\gamma}$ for each activity type $\gamma \in \Gamma$ (from the second to the second-to-last column), *iii)* $1 - \sum_{\gamma \in \Gamma} \phi_{\sigma,\gamma}$ (the last column). Note that only the 16/28 event types that have non-null θ_σ are shown.

3.2.3 Model formulation

In the proposed approach, a continuous-time stochastic model is constructed so as to fit the statistical characterization of the dataset. Activity recognition is then based on a measure of likelihood that depends on the probability that observed time-stamped events have in this model.

3.2.3.1 Syntax and semantics

The stochastic model is specified as a stochastic Time Petri Net (sTPN) [126]. As in [119], the formalism is enriched with flush functions which permit the marking of a set of places be reset to zero upon firing of a transition. This improves modeling convenience without any substantial impact on the complexity for the analysis.

Syntax An sTPN is a tuple $\langle P; T; A^-; A^+; A^\cdot; m_0; EFT; LFT; \mathcal{F}; \mathcal{C}; L\rangle$ where:

- P is the set of places;
- T is the set of transitions;
- $A^- \subseteq P \times T$, $A^+ \subseteq T \times P$, and $A^\cdot \subseteq P \times T$ are the sets of precondition, postcondition, and inhibitor arcs, respectively;
- $m_0 : P \rightarrow \mathbb{N}$ is the initial marking associating each place with a number of tokens;
- $EFT : T \rightarrow \mathbb{Q}_0^+$ and $LFT : T \rightarrow \mathbb{Q}_0^+ \cup \{\infty\}$ associate each transition with an *earliest* and a *latest firing time*, respectively, such that $EFT(t) \leq LFT(t)\ \forall\ t \in T$;
- $\mathcal{F} : T \rightarrow F_t^s$ associates each transition with a static Cumulative Distribution Function (CDF) with support $[EFT(t), LFT(t)]$;
- $\mathcal{C} : T \rightarrow \mathbb{R}^+$ associates each transition with a weight;

- $L : T \to \mathcal{P}(P)$ is a a *flush function* associating each transition with a subset of P.

A place p is termed an *input*, an *output*, or an *inhibitor* place for a transition t if $\langle p,t\rangle \in A^-$, $\langle t,p\rangle \in A^+$, or $\langle p,t\rangle \in A^{\cdot}$, respectively. A transition t is called *immediate* (IMM) if $[EFT(t), LFT(t)] = [0,0]$ and *timed* otherwise. A timed transition t is termed *exponential* (EXP) if $F_t(x) = 1 - e^{-\lambda x}$ over $[0,\infty]$ for some rate $\lambda \in \mathbb{R}_0^+$ and *general* (GEN) otherwise. A GEN transition t is called *deterministic* (DET) if $EFT(t) = LFT(t)$ and *distributed* otherwise. For each distributed transition t, we assume that F_t is absolutely continuous over its support and thus that there exists a Probability Density Function (PDF) f_t such that $F_t(x) = \int_0^x f_t(y)dy$.

Semantics The *state* of an sTPN is a pair $\langle m, \tau\rangle$, where $m : P \to \mathbb{N}$ is a marking and $\tau : T \to \mathbb{R}_0^+$ associates each transition with a time-to-fire. A transition is *enabled* if each of its input places contains at least one token and none of its inhibitor places contains any token; an enabled transition is *firable* if its time-to-fire is not higher than that of any other enabled transition. When multiple transitions are firable, one of them is selected to fire with probability $Prob\{t \text{ is selected}\} = \mathcal{C}(t)/\sum_{t_i \in T^f(s)} \mathcal{C}(t_i)$, where $T^f(s)$ is the set of firable transitions in s. When t fires, $s = \langle m, \tau\rangle$ is replaced by $s' = \langle m', \tau'\rangle$, where m' is derived from m by:

i) removing a token from each input place of t and assigning zero tokens to the places in $L(t) \subseteq P$, which yields an intermediate marking m_{tmp};

ii) adding a token to each output place of t.

Transitions enabled both by m_{tmp} and by m' are said *persistent*, while those enabled by m' but not by m_{tmp} or m are said *newly-enabled*; if t is still enabled after its own firing, it is regarded as newly enabled [18]. The time-to-fire of persistent transitions is reduced by the time elapsed in s, while the time-to-fire of newly-enabled transitions takes a random value sampled according to their CDF.

3.2.3.2 Structure and enhancement

Structure The model used to evaluate the likelihood of observed events is organized by composition of 7+1 submodels, which fit the observed behavior in the 7 activities classified in [125] and in the remaining *Idling* periods. Fig. 3.9 shows a fragment focused on *Idling* and *Preparing a beverage* activities.

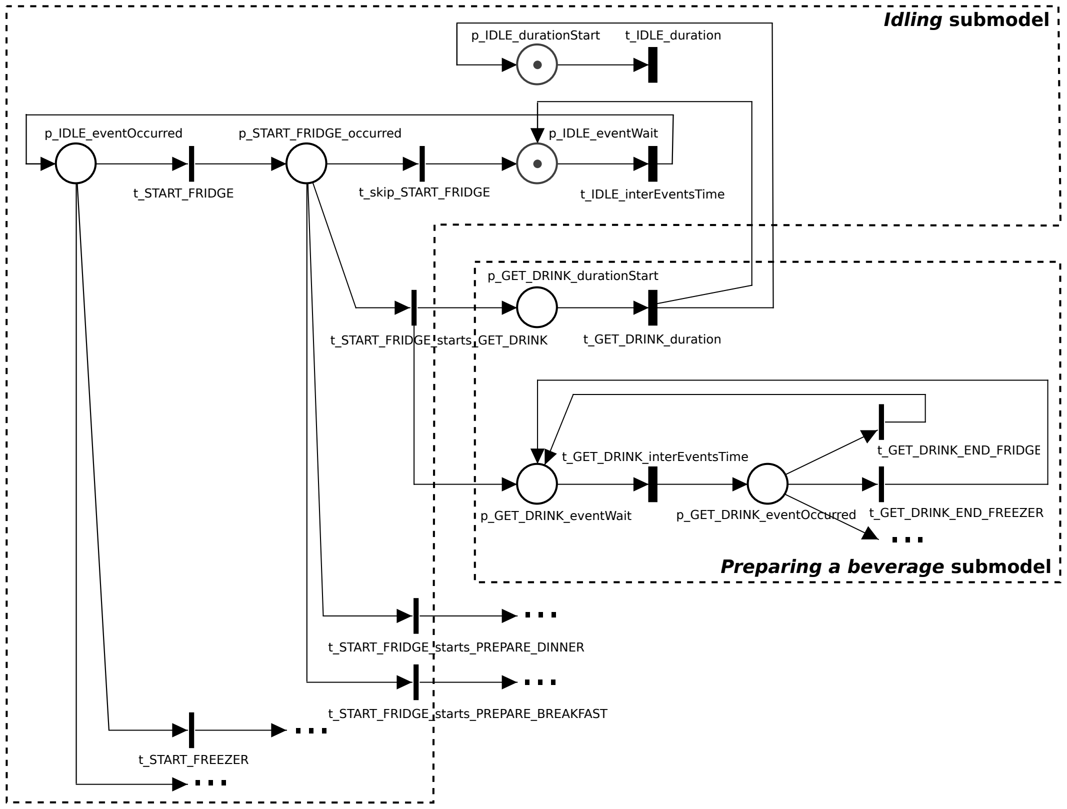


Figure 3.9. A fragment of the stochastic model used to evaluate the likelihood measure.

In the *Idling* submodel, places `p_IDLE_durationStart` and `p_IDLE_event Wait` receive a token each when the *Idling* starts, on completion of any activity. The token in `p_IDLE_eventWait` is removed whenever an event is observed (firing of the GEN transition `t_IDLE_interEventsTime`); in this case, different IMM transitions are fired depending on the type of the observed event (`t_START_FRIDGE`, `t_START_FREEZER`, ...), and then for each different type, a choice is made on whether the event is interpreted as a continuation of the *Idling* period (e.g., `t_skip_START_FRIDGE`) which will restore a token in `p_IDLE_eventWait`, or as the starting of each of the possible activities (e.g. `t_START_FRIDGE_starts_GET_DRINK`, `t_START_FRIDGE_starts_PREPARE_DINNER`,...). In parallel to all this, the token in `p_IDLE_durationStart` will be removed when the duration of *Idling* expires (at the firing of the GEN transition `t_IDLE_duration`) or when an observed event is interpreted as the beginning of any activity (e.g., at the firing of `t_START_FRIDGE_starts_GET_DRINK`); note that the latter case is not shown in the graphical representation and it is rather encoded in a flush function. Similarly, when the duration of *Idling* expires, the token in `p_IDLE_eventWait` will be removed by a flush function associated with transition `t_GET_IDLE_duration`.

In the *Preparing a beverage* submodel, places `p_GET_DRINK_durationStart` and `p_GET_DRINK_eventWait` receive a token each when an event observed during the *Idling* period is interpreted as the beginning of an instance of the *Preparing a beverage* activity. The token in `p_GET_DRINK_eventWait` is removed whenever an event is observed (firing of the GEN transition `t_GET_DRINK_interEventsTime`), and restored after the event is classified according to its type (IMM transitions `t_GET_DRINK_END_FRIDGE`, `t_GET_DRINK_END_FREEZER`, . . .). In parallel to this, the token in `p_GET_DRINK_durationStart` will be removed when the duration of the *Preparing a beverage* activity expires (at the firing of the GEN transition `t_GET_DRINK_duration`). Note that, in this case, also the token in `p_GET_DRINK_eventWait` will be removed: this is performed by a flush function associated with transition `t_GET_DRINK_duration`.

In so doing, in any reachable marking, only the submodel of the current activity contains two non-empty places, one indicating that the activity duration is elapsing (e.g., `p_GET_DRINK_durationStart`) and the other one meaning that the inter-events time is expiring (e.g., `p_GET_DRINK_eventWait`). Note that, as a naming convention, any transition named `t_EVENT` (where `EVENT` is an event type that may start an activity) or `t_ACTIVITY_EVENT` (where `EVENT` is an event type that may occur within `ACTIVITY`) accounts for an *observable* event, while all the other transitions correspond to *unobservable* events. Finally, note that the general structure of the model is open to modifications in various directions. For instance, in the submodels of activities, the choice between events might be easily made dependent on the duration of the inter-event time, which would allow a more precise classification without significantly impacting on the analyzability of the model. The viability of such evolutions mainly depends on the significance of the statistics that can be derived from the dataset.

Enhancement The actual topology of the model and its stochastic temporal parameters are derived in automated manner from the statistical indexes extracted in the abstraction of the dataset (Sect. 3.2.2).

The event types that can start an activity (e.g., in the model of Fig. 3.9, `t_START_FRIDGE_starts_GET_DRINK`, `t_START_FRIDGE_starts_PREPARE_DINNER`, ...) and the discrete probabilities in their random switch are derived from the *starting event type* statistic. The event types that can be observed during each activity (e.g., `t_GET_DRINK_END_FRIDGE`, `t_GET_DRINK_END_FREEZER`, ...) or can continue an *Idling* period and the discrete probabilities in their random switch are derived from the *event type* statistics.

The distribution associated with GEN transitions is derived from the *duration* statistic and the *inter-events time* statistic by fitting mean value μ and Coefficient of Variation CV as in [128]:

- if $0 \leq CV \leq \frac{1}{\sqrt{2}}$, we assume a *shifted exponential distribution*:

$$f(x) = \lambda e^{-\lambda(x-d)} \text{ over } [d, \infty),$$
$$\lambda = \sigma^{-1},$$
$$d = \mu - \sigma,$$
$$\text{where } \sigma^2 \text{ is the variance.}$$

- if $\frac{1}{\sqrt{2}} < CV < 1$, we use a *hypo-exponential distribution*:

$$f(x) = \lambda_1\lambda_2/(\lambda_1 - \lambda_2)(e^{-\lambda_2 x} - e^{-\lambda_1 x}) \text{ over } [0, \infty),$$
$$\lambda_i^{-1} = (\mu/2)(1 \pm \sqrt{2CV^2 - 1}), \text{ with } i = 1, 2.$$

- if $CV \approx 1$, we adopt an *exponential distribution*:

$$f(x) = \lambda e^{-\lambda(x)} \text{ over } [0, \infty),$$
$$\lambda = 1/\mu.$$

- if $CV > 1$, we consider a *hyper-exponential distribution*:

$$f(x) = \sum_{i=1}^{2} p_i \lambda_i e^{-\lambda_i x} \text{ over } [0, \infty),$$
$$p_i = [1 \pm \sqrt{(CV^2 - 1)/(CV^2 + 1)}]/2,$$
$$\lambda_i = 2 p_i \mu^{-1},$$
$$\text{with } i = 1, 2.$$

For instance, the duration and the inter-events time of *Preparing a beverage* are associated with a hyper-exponential distribution with parameters $p_1 = 0.773$, $p_2 = 0.227$, $\lambda_1 = 0.043358$, $\lambda_2 = 0.012717$, and $p_1 = 0.931$, $p_2 = 0.069$, $\lambda_1 = 0.243733$, $\lambda_2 = 0.017949$, respectively. In the sTPN model of Fig. 3.9, a GEN transition approximated by a hyper-exponential distribution is modeled as a switch between IMM transitions, each followed by an EXP transition (see Fig. 3.10). In a similar manner, a hypo-exponential distribution is represented by a sequence of EXP transitions, as depicted in Fig. 3.11, and a shifted exponential distribution is accounted by a sequence made of a DET transition and an EXP transition (see Fig. 3.12).

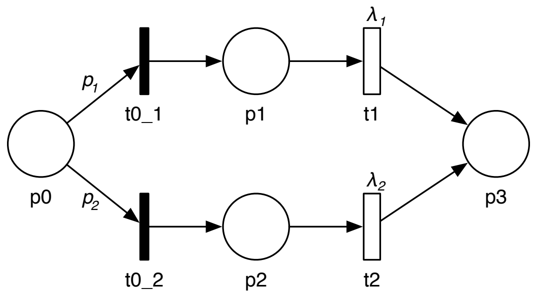


Figure 3.10. STPN formalism for hyper-exponential distribution.

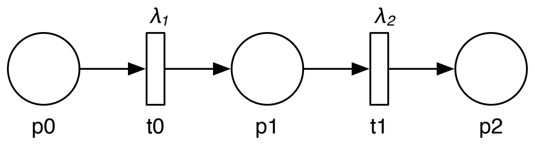


Figure 3.11. STPN formalism for hypo-exponential distribution.

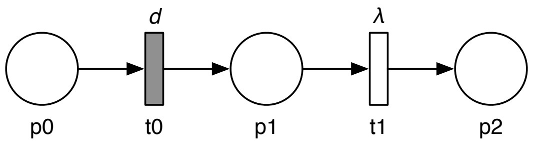


Figure 3.12. sTPN formalism for shifted-exponential distribution.

3.2.4 Online classification of ADLs

We use the transient probability $\pi_\gamma(t)$ that an activity of type $\gamma \in \Gamma$ is being performed at time t as a *measure of likelihood* for γ at t. A classification $\mathcal{P}(t)$ emitted at time t consists of the set of activity types that may be performed at t, each associated with the likelihood measure, i.e., $\mathcal{P}(t) = \{\langle\gamma, \pi_\gamma(t)\rangle \mid \pi_\gamma(t) \neq 0\}$. Between any two subsequent events $e_n = \langle\sigma_n, t_n\rangle$ and $e_{n+1} = \langle\sigma_{n+1}, t_{n+1}\rangle$, a classification $\mathcal{P}(t)$ is emitted at equidistant time points in the interval $[t_n, t_{n+1}]$, i.e., $\forall\ t \in \{t_n,\ t_n + q,\ t_n + 2q,\ \ldots,\ t_{n+1}\}$, with $q \in \mathbb{R}^+$. In the experiments, we assume the activity type with the highest measure of likelihood as the classified class to be compared against the actual class annotated in the ground truth, i.e., at time t, the classified activity is $\gamma \mid \pi_\gamma(t) = \max_{a\in\Gamma|\langle a,\pi_a(t)\rangle\in\mathcal{P}(t)}\{\pi_a(t)\}$.

As a result of the prescribed model structure and the specific enhancement, the stochastic model subtends a Markov Regenerative Process (MRP) [34, 35, 22] under enabling restriction, i.e., no more than one GEN transition is enabled in each marking (only the duration of four activities is modeled by a shifted exponential distribution, thus no more than one DET transition is enabled in each marking). According to this, online classification can be performed by leveraging the regenerative transient analysis of [51]. The solution technique of [51] samples the MRP state after each transition firing, maintaining an additional timer τ_{age} accounting for the absolute elapsed time; each sample, called *transient class*, is made of the marking and the joint PDF of τ_{age} and the times-to-fire of the enabled transitions. Within a given time limit T, enumeration of transient classes is limited to the first regeneration epoch and repeated from every regeneration point (i.e., a state where the future behavior is independent from the past), enabling the evaluation of transient probabilities of reachable markings through the solution of generalized Markov renewal equations.

In the initial transient class of the model, the marking assigns a token to places `p_IDLE_durationStart` and `p_IDLE_eventWait`, all transitions are newly enabled, and τ_{age} has a deterministic value equal to zero. After n observed events $e_1 = \langle \sigma_1, t_1 \rangle$, ..., $e_n = \langle \sigma_n, t_n \rangle$, let $S_n = \{\langle s_n^i, \omega_n^i \rangle\}$ be the set of possible transient classes s_n^i having probability ω_n^i, where $\sum_{i|\langle s_n^i, \omega_n^i \rangle \in S_n} \omega_n^i = 1$. Regenerative transient analysis [51] of the model is performed from each possible transient class $\langle s_n^i, \omega_n^i \rangle \in S_n$ up to any observable event within a given time limit, which is set equal to 48 h to upper bound the time between any two subsequent events. This allows one to evaluate transient probabilities of reachable markings, i.e., $p_m^{n,i}(t) = Prob\{M^{n,i}(t) = m\}\ \forall\ t \leq T,\ \forall\ m \in \mathcal{M}^{n,i}$, where $\mathbb{M}^{n,i} = \{M^{n,i}(t), t \geq 0\}$ is the underlying marking process, and $\mathcal{M}^{n,i}$ is the set of markings that are reachable from s_n^i. Since in any reachable marking only the submodel of the ongoing activity contains non-empty places, transient probabilities of markings are aggregated to derive transient probabilities of ongoing activities $\pi_\gamma(t) = \sum_{\langle s_n^i, \omega_n^i \rangle \in S_n} \omega_n^i \sum_{m \in \mathcal{M}_\gamma^{n,i}} p_m^{n,i}(t)\ \forall\ \gamma \in \Gamma$, where $\mathcal{M}_\gamma^{n,i}$ is the set of markings that are reachable from s_n^i and have non-empty places in the submodel of the activity type γ.

Whenever an event $e_{n+1} = \langle \sigma_{n+1}, t_{n+1} \rangle$ is observed, any tree of transient classes enumerated from class $\langle s_n^i, \omega_n^i \rangle \in S^n$ is explored to determine the possible current classes and their probability. More specifically:

- the possible current classes are identified as those classes that can be reached after a time $t_{n+1} - t_n$ through a sequence of unobservable events followed by the observed event e_{n+1};

- any possible current class s^j_{n+1} is a regeneration point since, by model construction, each GEN transition is either newly enabled or enabled by a deterministic time (i.e., the timestamp $t_{n+1} - t_n$);
- the probability ω^j_{n+1} of s^j_{n+1} is obtained as $\lim_{t \to t^-_{n+1}} \zeta_{s^p_{n+1}}(t) \cdot \rho$, where s^p_{n+1} is the last class where the model waits for the arrival of the next event e_{n+1}, $\zeta_{s^p_{n+1}}(t)$ is the probability of being in class s^p_{n+1} at time t, and ρ is the product of transition probabilities of the arcs encountered from s^p_{n+1} to s^j_{n+1};
- in the limit case that s^p_{n+1} is vanishing, ω^j_{n+1} is obtained as the product of transition probabilities of the arcs encountered from the root class to s^j_{n+1}.

Hence, the approach is iterated, performing transient analysis from any new current class up to any observable event, still encountering regeneration points after each observed event.

By construction, the approach complexity is linear in the number of observed events. For each observed event $e_n = \langle \sigma_n, t_n \rangle$, the number of transient trees to enumerate is proportional to the number of possible parallel hypotheses $|\mathcal{P}(t_n)|$, i.e., the number of activity types that may be performed at time t_n; moreover, the depth of each transient tree is proportional to the number of events that may occur between two observations (which is bounded in the considered application context), and relatively insensitive to the time elapsed between observed events.

3.2.5 Computational experience

Applicability of the proposed approach was validated by experimenting on a reference annotated dataset [125].

3.2.5.1 Experimental setup

Experiments were performed on the dataset [125], using a *dual change-point* representation for sensor events as detailed in Sect. 3.2.1. We split data provided by the computed statistics and event logs into training and test sets using a *Leave One Day Out* (LOO) approach, which consists of using each instance of one full day sensor readings for testing and the instances of the remaining days for training. Since, in each test, classifications are emitted starting from the first observed event of the day, we extended online analysis up to the first event of the next day. To avoid inconsistencies in the characterization of *Leaving house*, during which all the event types were observed, we removed from the training sets all events occurring during *Leaving house* that are not of type *start_front door* and *end_front door*. Moreover, whenever the ground truth includes concurrent activities, we considered our classification correct if the classified activity type is equal to any of the concurrent activity types.

We performed experiments using two fitting techniques in the evaluation of the duration and the inter-events time statistics:

- only exponential distributions (i.e., *exponential* case);
- different classes of distributions based on the CV value, as discussed in Sect. 3.2.3.2 (i.e., *non-Markovian* case).

On a machine with an Intel Xeon 2.67 GHz and 32 GB RAM, the evaluation for a single day took on average 43 s for the exponential case and 18 minutes for the non-Markovian case.

We evaluated the performance of our approach computing, for each activity class, three measures, derived from the number of true positives (TP), false positives (FP), and false negatives (FN):

- *Precision*

  $$Pr = TP/(TP + FP),$$

  which accounts for the accuracy provided that a specific class has been classified;

- *Recall*

  $$Re = TP/(TP + FN),$$

  which represents the ability to select instances of a certain class from a dataset;

- *F-measure*

  $$F_1 = 2 * Pr * Re/(Pr + Re),$$

  which is the harmonic mean of precision and recall.

We also compared the outcome of our experiments with the results reported in [125], obtained using a generative model (i.e., an HMM) and a discriminative one (i.e., a CRF) in combination with *offline* inference and the change-point representation. To make this comparison possible, we sampled the result of our classification using a time-slice of duration $\Delta t = 60\ s$ and we considered two additional measures:

- *Accuracy*

  $$A = 1/N \sum_{i=1}^{N_c} TP_i,$$

  which is the average percentage of correctly classified time-slices, with N being the total number of time-slices, N_c the total number of activity classes, and TP_i the number of TP of class i;

- *Average recall*

  $$\bar{Re} = 1/N_c \sum_{i=1}^{N_c} Re_i,$$

  which is the average percentage of correctly classified time-slices per class, with Re_i being the recall of class i.

3.2.5.2 Results

Table 3.4 shows the confusion matrix for the exponential and the non-Markovian cases, which reports in position i, j the number of time-slices of class i classified as class j; Table 3.5 shows precision, recall and F_1 score as derived from the confusion matrix. Results show that *Idling*, *Leaving house*, and *Sleeping* are the activities with the highest F_1 score. In terms of F_1 score, the non-Markovian case outperforms the exponential one for all activity classes except for *Preparing a beverage* and *Taking shower*. In terms of precision, the non-Markovian case performs worse only for *Sleeping*. Conversely, in terms of recall, the exponential case outperforms the non-Markovian one for *Preparing a beverage*, *Preparing breakfast*, *Taking shower*, and *Toileting*, and performs worse for *Idling*, *Preparing dinner*, and *Sleeping*. Note that the precision, recall, and F_1 score of *Leaving house* are identical in both cases.

Accuracy and average recall are summarized in Table 3.6, and compared with results from [125]. As we can see, fitting statistical data according to the CV (non-Markovian case), we achieve the highest accuracy, both with respect to our exponential case and with respect to HMM and CRF. Nevertheless, the exponential case, HMM, and CRF outperform the non-Markovian case in terms of average recall.

3.2.5.3 Discussion

Experimentation developed so far achieves results that compare well with the HMM and CRF approaches, with a slight increase in precision and a slight reduction in recall. The proposed approach is open to various possible developments, and the insight on observed cases of success and failure comprises a foundation for refinement and further research on which we are presently working.

In the present implementation, classification of the current activity relies only on past events, which is for us instrumental to open the way to the integration of classification with online prediction. However, the assumption of this causal constraint severely hinders our approach in the comparison against

	Idling	**Leaving house**	**Preparing a beverage**	**Preparing breakfast**	**Preparing dinner**	**Sleeping**	**Taking shower**	**Toileting**
Idling	2975/3471	330/330	37/6	82/16	733/400	588/658	131/65	98/28
Leaving house	184/209	22219/22219	1/0	0/0	101/56	25/71	22/4	15/8
Preparing a beverage	8/6	1/1	5/2	0/0	6/11	1/1	0/0	0/0
Preparing breakfast	5/14	0/0	3/2	39/24	15/22	8/8	0/0	0/0
Preparing dinner	83/107	0/0	4/1	39/13	214/221	0/0	0/0	1/0
Sleeping	194/215	0/0	0/0	0/0	0/0	11226/11430	3/1	231/8
Taking shower	94/100	0/0	0/0	0/0	1/1	17/41	105/79	4/0
Toileting	46/60	2/2	0/0	0/0	0/1	36/52	7/8	66/33

Table 3.4. Confusion matrix showing the number of timeslices of each class i (first column) classified as class j (other columns): exponential case/non-Markovian case. Diagonal elements represent TP, while FN (FP) can be read along rows (columns).

	Exponential			**non-Markovian**		
	Precision	**Recall**	**F_1**	**Precision**	**Recall**	**F_1**
Idling	82.89	59.81	69.49	83.00	69.78	75.82
Leaving house	98.52	98.46	98.49	98.52	98.46	98.49
Preparing a beverage	10.00	23.81	14.09	18.18	9.52	12.50
Preparing breakfast	24.38	55.71	33.91	45.28	34.29	39.02
Preparing dinner	20.00	62.76	30.33	31.04	64.62	41.94
Sleeping	94.33	96.33	95.32	93.22	98.08	95.59
Taking shower	39.18	47.51	42.95	50.32	35.75	41.80
Toileting	15.90	42.04	23.08	42.86	21.15	28.33

Table 3.5. Precision, recall, and F_1 score achieved for each activity type.

	Accuracy	**Average recall**
Exponential case	**92.11**	**60.80**
non-Markovian case	**93.69**	**53.96**
HMM [125]	80.00	67.20
CRF [125]	89.40	61.70

Table 3.6. Accuracy and average recall achieved by the exponential and non–Markovian cases (dual change-point representation and online analysis), compared with those reported in [125] for HMM and CRF (change-point representation and offline analysis).

the offline classification implemented in [125] through HMM and CRF. Online classification results reported in [125] are unfortunately not comparable due to the different abstraction applied on events, and it should be remarked that in any case they are not completely online as the classification at time t relies on all the events that will be observed within the end of the timeslice that contains t itself, which makes a difference for short duration activities. For the purposes of comparison, our online approach can be relaxed to support offline classification by adding a backtrack from the final states reached by the classificator. This should in particular help the recall of short activities started by events that can be accepted also as the beginning of some longer activity (e.g., *Preparing a beverage* with respect to *Preparing dinner*). We also expect that this should reduce the number of cases where a time period is misclassified as *Idling*.

The statistics of durations are now fitted using the basic technique of [128] which preserves only mean value and coefficient of variation. Moreover, the deterministic shift introduced in the approximation of hypo-exponential distributions with low CV causes a false negative for all the events occurring before the shift completion, which is actually observed in various cases. Approximation through acyclic Continuous PHase type (CPH) distributions [81, 50, 104] would remove the problem and allow an adaptable trade-off between accuracy and complexity. In particular, a promising approach seems to be the method of [21] which permits direct derivation of an acyclic phase type distribution fitting not only mean value and coefficient of variation but also skewness.

Following a different approach, the present implementation is completely open to the usage of any generally distributed (GEN) representation of activity durations. This would maintain the underlying stochastic process of the online model within the current class, i.e., Markov Regenerative Processes (MRP) that run under enabling restriction [35] and guarantee a regeneration within a bounded number of steps. In this case, online classification could be practically implemented using various tools, including Oris [51, 25], TimeNet [135] and GreatSPN [7].

In the present implementation, classification is unaware of the absolute time, which may instead become crucial to separate similar activities, such as for instance *Preparing breakfast* and *Preparing dinner*. To overcome the limitation, the model should in principle become non-homogeneous, but a good approximation can be obtained by assuming a discretized partition of the daytime, which can be cast in the online model as a sequence of deterministic delays. By exploiting the timestamps, at each event the current estimation of the absolute time is restarted. Under the fair assumption that at least one event is observed in each activity, the underlying stochastic process of the on-line model still falls in the class of MRPs that encounter a regeneration within a bounded number of steps, and can thus be practically analyzed through the Oris Tool [51, 25].

Chapter 4

Performance engineering of meta-modeling architectures

This Chapter explores meta-modeling architectures from a performance point-of-view, investigating the performance impact that new persistence approaches based on promising NoSQL technologies [115, 2, 3] can bring in the model-driven re-engineering of a meta-modeling architecture with respect to consolidated relational solutions.

4.1 Relational solutions and inefficiencies

Due to the high degree of abstraction of the underlying meta-model, a meta-modeling architecture is often exposed to major performance inefficiencies, in terms of extra processing and instantiation, at run-time, of an increased number of objects (and relationships) for describing the whole domain. These drawbacks are largely exacerbated when the domain logic is persisted over a relational storage layer, due to the nature of the domain model and its mismatch with the relational tier [6].

These performance issues, that translate in longer time required for key persistence operations, can be partially mitigated with ad-hoc optimizations in the design of the relational database [110], including: the choice of a particular representation for class inheritance; the use of auxiliary tables to store additional information; and the smart use of data fetching.

The interposition of an Object-Relational Mapping (ORM) layer between the domain layer and the data layer can mitigate this problem. In the practice of development of Java enterprise applications, the Java Persistence API (JPA) specification represents a mature and state-of-the-art ORM solution which grants many benefits [20]. First of all, it allows to persist domain classes with a minimal boilerplate code, thanks to simplified annotation facilities. Moreover, it provides full integration with the Java application stack, composed by other technologies such as EJB (for encapsulating the business logic) and CDI (for implementing the *Inversion of Control* pattern [76]). However, JPA further increases the degree of indirection and this can have negative effects on system performance, also due to the loss of design control on the impact that domain logic operations have on the storage process.

4.2 Towards NoSQL technologies

With the rise of the *Not Only SQL (NoSQL)* movement [115], other options in the storage modeling are now available, providing various advantages.

On the one hand, the high degree of flexibility provided by NoSQL solutions gives space to alternative choices in the definition of the storage data model, which is, to a large extent, independent from the structure of object types. The absence of a fixed schema provides multiple options concerning the definition of the database structure, facilitating the representation of heterogeneous data characterized by high variability over time, such as in the case of meta-modeling architectures.

On the other hand, while relational databases are not able to scale out easily on commodity clusters preferring scale up policies as workload increases, new NoSQL databases are designed to horizontal scaling by adding new nodes, leading to a overall improvement in the data access time [109]. This motivates the investigation on engineering the performance of working systems by changing the storage schema from a relational + ORM persistence stack to a NoSQL solution, while preserving the domain logic structure.

In the rest of this Section, we compare and experiment different data models implemented over two distinct NoSQL persistence layers, i.e. a graph-oriented database called *Neo4j* [3], and a document-oriented database called *MongoDB* [2], applied to the case of a meta-modeling architecture originally persisted over a MySQL + Hibernate technology stack, i.e. the *Empedocle* EHR system (see Chapter 2 for more details). While referring to this case for the sake of experimentation concreteness, most of the subsequent discussion about the development of a graph-oriented or document-oriented database representation as well as about the impact that NoSQL technologies can have on system performance are more generally applicable to most schemes that can be designed in the style of a meta-modeling architecture.

4.2.1 A data model for Neo4j

Neo4j [3] relies on a *graph-oriented* structure, which can natively represent the domain logic of a meta-modeling architecture whose data structures are direct acyclic graphs and trees (see Fig. 2.4) [127].

As a schema-less database, the data model in Neo4j is inherently defined by the *nodes* and *relationships* persisted in the database. Every node and relationship can also be characterized by an arbitrary number of *properties*. From version 2.0, Neo4j developers tweaked its schema-less nature by introducing *labels* and *indexes*, two concepts that help modeling data in a more organized way, without losing the database original adaptability. Specifically, labels can be used to group together nodes, and each node can optionally be labeled with one or more text descriptions, and indexed to improve query expressiveness and flexibility. Moreover, indexes can be defined on properties of labeled nodes, to improve performance during query operations, similarly to the relational case. Both labels and indexes are optional.

In the concrete case of the *Empedocle EHR system*, modeling the domain logic of Fig. 2.3 in Neo4j comes down to:

1. identifying the node structure that characterize the model;
2. defining relationships between nodes;
3. defining properties that characterize nodes and relationships;
4. labeling with appropriate qualifiers.

Following these steps, as depicted in the schema of Fig. 4.1, each class that is an entity in the original model has been represented as a node in the target model, and labeled with correspondent qualifiers. In so doing, nodes that belong to `Fact` or `FactType` hierarchy are qualified using two labels: the first one to identify the hierarchy, and the second one to define their role in the class hierarchy (e.g., *Fact:QualitativeFact* qualifies a `QualitativeFact` inside a `Fact` hierarchy).

As it can be observed in the schema, the *name* property is used for identifying, at the knowledge level, a *named* `FactType`. The *value* property is used to record, at the operational level, the value assumed by a `TextualFact` or a `QuantitativeFact` node.

Another characteristic of the graph model is the capability of modeling `TypeLink` and `FactLink` classes using relationships. These two classes were introduced in the original model to represent the parent-child association between `FactType` or `Fact` classes. For this reason, they can be naturally modeled as a relationship in a graph-oriented model. In addition, since Neo4j represents relationships as directed arcs that can be traversed in both directions, this allows to simplify the model introducing a single relationship, called *HasChildren*, for modeling `TypeLink` and `FactLink` classes, without any impact on query capabilities.

Since Neo4j allows to put a relationship only between two nodes, this precludes the possibility to use a relationship to represent the existing reference

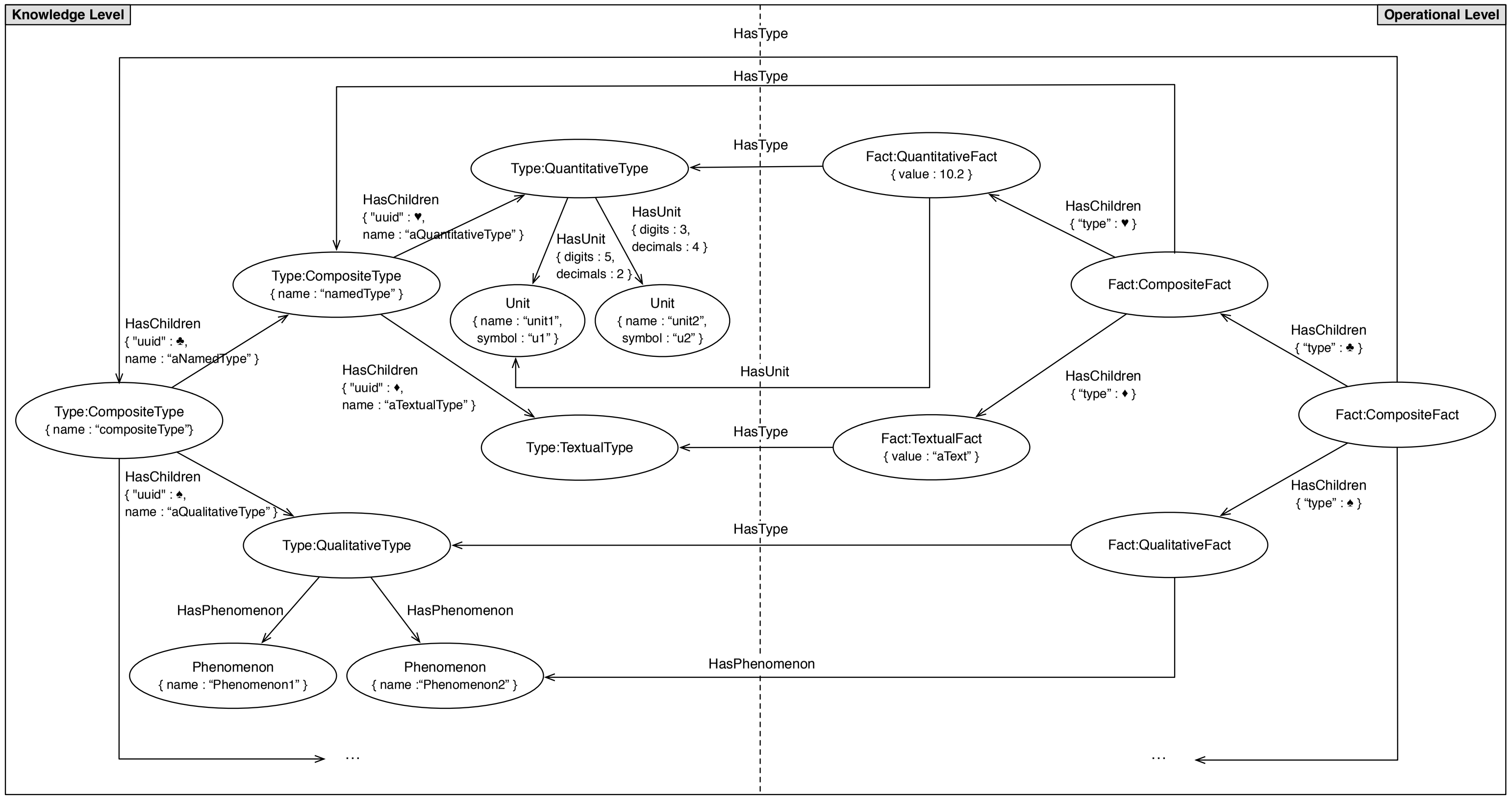

Knowledge Level
Operational Level
HasType
HasType
HasType
Type:QuantitativeType
Fact:QuantitativeFact { value : 10.2 }
HasChildren { "uuid" : ♥, name : "aQuantitativeType" }
HasUnit { digits : 3, decimals : 4 }
HasUnit { digits : 5, decimals : 2 }
HasChildren { "type" : ♥ }
Type:CompositeType { name : "namedType" }
Unit { name : "unit1", symbol : "u1" }
Unit { name : "unit2", symbol : "u2" }
Fact:CompositeFact
HasChildren { "uuid" : ♣, name : "aNamedType" }
HasChildren { "type" : ♣ }
HasChildren { "uuid" : ♦, name : "aTextualType" }
HasUnit
HasChildren { "type" : ♦ }
Type:CompositeType { name : "compositeType"}
Fact:CompositeFact
Type:TextualType
HasType
Fact:TextualFact { value : "aText" }
HasChildren { "uuid" : ♠, name : "aQualitativeType" }
HasChildren { "type" : ♠ }
Type:QualitativeType
HasType
Fact:QualitativeFact
HasPhenomenon
HasPhenomenon
Phenomenon { name : "Phenomenon1" }
Phenomenon { name :"Phenomenon2" }
HasPhenomenon
...
...

Figure 4.1. An instance of the domain model of the *Empedocle* EHR system as represented on the graph-oriented model of Neo4j. While oval shapes represent nodes, arcs between nodes represent relationships. Labels are written in bold, and properties are reported between braces. For reasons of readability, *uuid* property values have been replaced by symbols.

between `TypeLink` and `FactLink`, as in the original model. Properties have been used to solve this problem, as follows:

- the *uuid* property of each `TypeLink` is used to store an identifier value;
- the same value is copied into the *type* property of the related `FactLink`.

Finally, `Fact` and `FactType` nodes are linked together via the *HasType* relationship. The self-explanatory *HasUnit* and *HasPhenomenon* relationships are ambivalent across the knowledge and the operational level, and are used to connect:

- a `QuantitativeType` node with a set of possible `Unit` nodes, and the corresponding `QuantitativeFact` node with the selected `Unit` node;
- a `QualitativeType` node with a set of possible `Phenomenon` nodes, and the corresponding `QualitativeFact` node with the selected `Phenomenon` node.

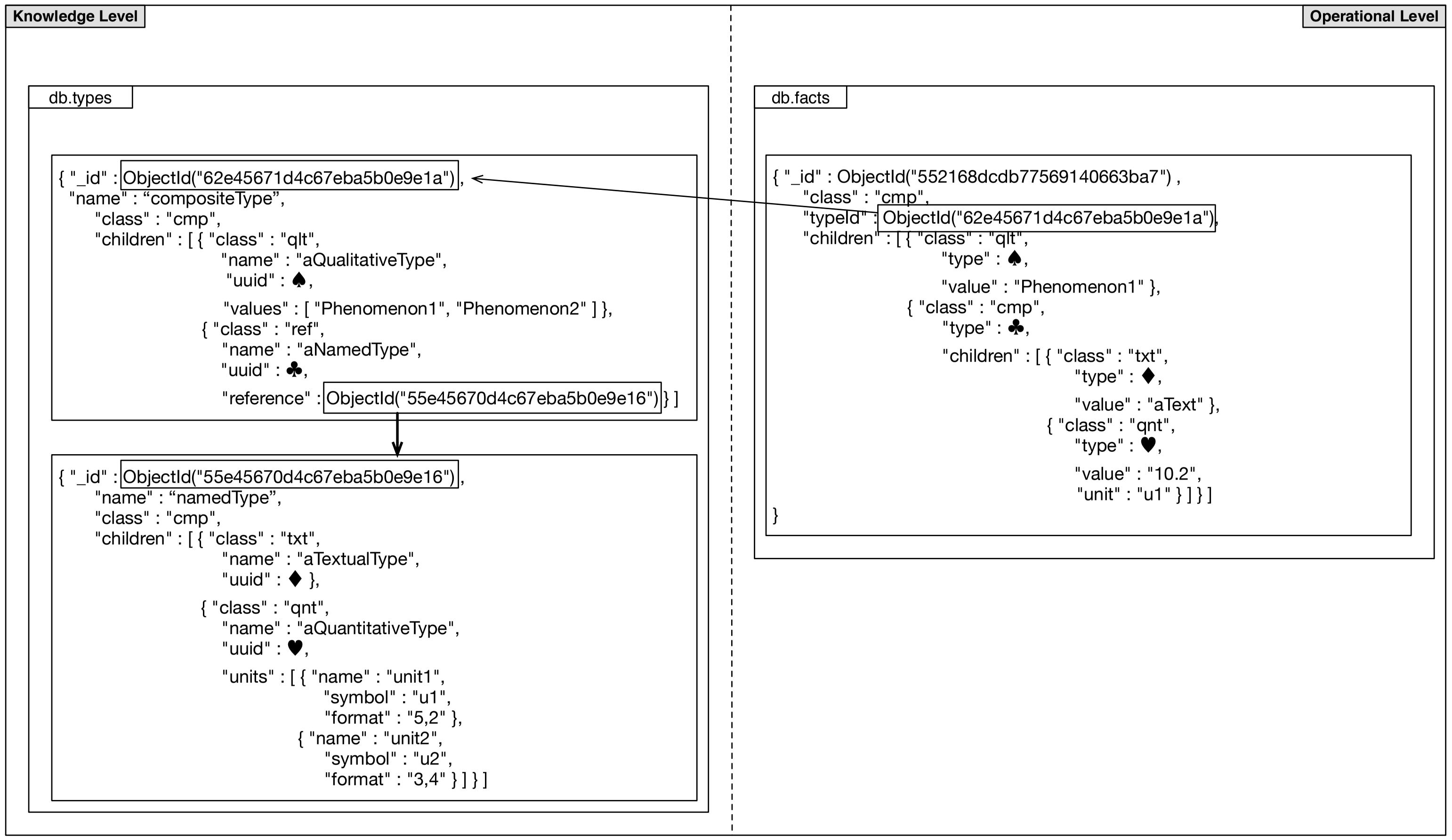
4.2.2 A data model for MongoDB

MongoDB [2] data model is based upon a *document-oriented* structure. A document is a collection of attribute-value pairs, with values that can be basic types, array of values or nested sub-documents. Documents with similar characteristics are grouped together and stored in collections. Relations between documents can be represented in two ways:

- using *references*, that produce a normalized data model;
- by *embedding* related data in documents, producing denormalized models.

In particular, the use of denormalization techniques [63] is promoted by document-oriented NoSQL solutions for discouraging the use of JOIN queries, and solving typical performance issues that affect relational databases.

The schema of Fig. 4.2 illustrates how the domain logic of the *Empedocle* EHR system is represented on the document-oriented model of MongoDB.



```
{ "_id" : ObjectId("62e45671d4c67eba5b0e9e1a"),
  "name" : "compositeType",
    "class" : "cmp",
    "children" : [ { "class" : "qlt",
                     "name" : "aQualitativeType",
                     "uuid" : ♠,
                     "values" : [ "Phenomenon1", "Phenomenon2" ] },
                   { "class" : "ref",
                     "name" : "aNamedType",
                     "uuid" : ♣,
                     "reference" : ObjectId("55e45670d4c67eba5b0e9e16") } ]
```

```
{ "_id" : ObjectId("55e45670d4c67eba5b0e9e16"),
    "name" : "namedType",
    "class" : "cmp",
    "children" : [ { "class" : "txt",
                     "name" : "aTextualType",
                     "uuid" : ♦ },
                   { "class" : "qnt",
                     "name" : "aQuantitativeType",
                     "uuid" : ♥,
                     "units" : [ { "name" : "unit1",
                                   "symbol" : "u1",
                                   "format" : "5,2" },
                                 { "name" : "unit2",
                                   "symbol" : "u2",
                                   "format" : "3,4" } ] } ]
```

```
{ "_id" : ObjectId("552168dcdb77569140663ba7") ,
    "class" : "cmp",
    "typeId" : ObjectId("62e45671d4c67eba5b0e9e1a"),
    "children" : [ { "class" : "qlt",
                     "type" : ♠,
                     "value" : "Phenomenon1" },
                   { "class" : "cmp",
                     "type" : ♣,
                     "children" : [ { "class" : "txt",
                                      "type" : ♦,
                                      "value" : "aText" },
                                    { "class" : "qnt",
                                      "type" : ♥,
                                      "value" : "10.2",
                                      "unit" : "u1" } ] } ]
}
```

Figure 4.2. An instance of the domain model of the *Empedocle* EHR system as represented on the document-oriented model of MongoDB. The two sides of the Figure show the collections used to persist `FactType` and `Fact` instances, named *db.types* and *db.facts*, respectively. At knowledge level, two *named* types are depicted: a *compositeType* and a *namedType*. The first type refers to the second one, as highlighted by the use of the *ObjectId* reference. Both types include *anonymous* types as embedded documents. For reasons of readability, *uuid* property values have been replaced with symbols.

Usually, modeling an object-oriented domain logic via a document-based data model can be achieved in a direct way. Unfortunately, this simplicity is weakened when dealing with meta-modeling architectures, due to the indirect structure of the model.

The proposed solution attains a good balance, mixing together documents embedding approaches with references techniques [63] for obtaining a flexible data representation without performance degradation. In so doing, the `FactType` hierarchy comprises a typical example of mixed modeling. Specifically, while *named* `FactType` instances are persisted as distinct documents, and referenced by other documents using the *ObjectId* identifier, *anonymous* `FactType` instances are persisted as embedded documents inside the named `FactType` document in which they are defined.

To efficiently recognize the subtyping-class of an instance in the `FactType` or `Fact` hierarchy, every persisted document has a property called *class* that can assume the following values: *txt*, for referring to a `TextualType` or `TextualFact` instance; *qlt*, for referring to a `QualitativeType` or `QualitativeFact` instance; *qnt*, for referring to a `QuantitativeType` or `QuantitativeFact` instance; and, *cmp*, for referring to a `CompositeType` or `CompositeFact` instance. In so doing, it is sufficient to check the *class* property value of a document to recognize its nature, avoiding to pre-emptively explore its properties. In the case of referring to a *named* `FactType`, the *class* property assumes the *ref* value, and an additional property called *reference* contains the *ObjectId* of the referenced *named* `FactType`.

The different behavior used for persisting `FactType` instances drops the need to persist `TypeLink` instances as separate entities. For this reason, `TypeLink` and `FactType` classes are modeled in MongoDB as a single entity, and the *name* property of embedded documents inside `CompositeType` instances corresponds to the `TypeLink` *name* property of the original model. Note that, since embedded documents are always *anonymous*, the `FactType` *name* property is specified only for the root document of a `FactType` hierarchy.

The `Fact` hierarchy does not have the same need of reusability and referencing that characterize `FactTypes`. For this reason, `Fact` instances can always be represented as a single document, in which `Fact` children are embedded as sub-documents. In so doing, the number of queries for data retrieval is considerably limited.

`Fact` and related `FactType` instances are linked together with different strategies, depending on the role of the `Fact` in the hierarchy. In the case of a `Fact` root document, the *typeId* property stores the *ObjectId* of the referenced `FactType` instance. Otherwise, when dealing with sub-documents of the `Fact` root, a *type* property refers to the *uuid* value of the corresponding `FactType`. Consequently, for completely retrieving a `Fact` and its related `FactType`, we need to:

1. query for the `Fact` root (with children as embedded documents);
2. query for the related `FactType` using the *ObjectId* referenced by the *typeId* property of the `Fact` root;
3. link together retrieved `Fact` children and `FactType` children instances using *uuid* and *type* properties.

For the sake of completeness, `Phenomenon` entities are modeled as embedded documents inside `QualitativeFact` and `QualitativeType` documents with the intent of minimizing the number of retrieval query in reading operations. In the same manner, `Unit` entities are modeled inside `QuantitativeFact` and `QuantitativeType` documents.

4.2.3 The information equivalence problem

A comparison of the performance among different data storage implementations (i.e. from relational to a document- or graph-oriented model) requires that they are in some sense equivalent. Since data can be modeled in various ways, through the use of different data structures offering the same *information capacity*, a notion of model equivalence, or hierarchy of equivalences [52], is required to be defined.

In a general sense, two data structures can be considered equivalent in terms of information-capacities if they can be associated to the same number of states, such that each state of a data structure can be mapped to a *database state* of the other structure, preserving any relationship attribute value.

For the purpose of our experimentation, it is not necessary to prove the *complete* equivalence between two representations, but it is sufficient to prove the *query* equivalence of two models [9], i.e. the possibility to extract the same information from both models through query operations. Specifically, the equivalence problem consists in casting information data into structures (i.e. graphs or tree) of the same type.

Comparing and matching graphs is a well-known NP-complete problem [45], and different approaches have been proposed to determine the distance between two graphs using specific heuristic [30, 37].

In our case, proving the equivalence of *Neo4j* and *MongoDB* data models with respect to the actual relational model means showing that they have the same representativeness of information. This means that the equivalence problem will be focused on showing that two data structures are exactly identical in terms of represented information, rather than identifying similarities and differences between data models. Moreover, it is not necessary to check the *query dominance* for the new data models; it is sufficient to show that it is possible to query the same structure across different representations.

In a practical manner, we consider equivalent two data representations of the same domain logic using different persistence models when the carried information can be serialized into an equivalent string of information. In so doing, given two different persistence models, named A and B, A and B are equivalent if it is possible to generate the same string serialization for each given `Examination` and `ExaminationType` instance represented in A and B. Consequently, if A is a valid model, and A and B are equivalent, than B is also valid. Note that we assume that the actual relational model is a valid reference model.

In order to prove the validity of converted NoSQL models, we started by choosing a dataset with an arbitrary number of clinical information data, persisted in the relational model. Then, we have retrieved all `Examinations` and `ExaminationTypes` instances contained in the dataset, and we have serialized the information data in a string representation. Finally, for testing the equivalence, we have converted information data from the relational model to the target NoSQL model, serializing again the information data, and comparing the resulting string with the string obtained from the relational model at the previous step. The validation process is considered successful, if we are able to obtain an equivalence between the reference relational model and the target NoSQL model for every string of information.

Fig. 4.3 illustrates an example of the string produced during the serialization process applied to the information data as represented using the models depicted in Figs. 4.1 and 4.2.

```
"compositeType" : {
  "aNamedType" : {
    "aQuantitativeType" : "10.2␣u1"
    "aTextualType" : "aText"
  }
  "aQualitativeType" : "Phenomenon2"
}
```

Figure 4.3. An example of serialization of a clinical `Examination`. The pattern used to serialize the information is as follows: `type.name : fact.value`. The structure of the serialization is deliberately similar to a JSON document, due to its simple and readable syntax.

4.2.4 Computational experience

An experimentation was carried out to evaluate the performance of three different implementations based on MySQL + Hibernate, Neo4j, and MongoDB, and their sensitivity to the characteristics of the dataset.

4.2.4.1 Experimental setup

We can expect that the response time of different storage schemes be dependent on the complexity of the collection of domain logic objects that are read-from or written-to the persistence layer. Due to the pattern-based architecture of classes in the domain logic, objects are organized in an almost tree-like structure, and their complexity can thus be characterized in terms of nodes and depth of the tree. For this reason, we experimented with two different kind of datasets:

- a *real* dataset of clinical examinations acquired in the *Empedocle* EHR system for which we provide a description of the statistics about the number of nodes and the depth of the tree structure;
- a *synthetic* dataset for which we can control the statistics so as to stress the indexes of complexity.

The *real* dataset consists of about 13 000 examinations that belong to the same speciality and thus share the same structure. The dataset was conveniently anonymized by omitting patients' personal information, and by obfuscating textual observations recorded during each clinical session.

Table 4.1 summarizes the complexity of the examination structure, i.e. the number of `FactTypes` included in each examination. The structure of the examination includes 243+110+99 fields, which are organized in a graph whose depth (intended as the maximum distance from the root node) is equal to 8, and which includes 144 `FactTypes` that act as composition nodes.

Depth	8	
Number of nodes	596	
	CompositeType	144
	QualitativeType	243
	QuantitativeType	110
	TextuaType	99

Table 4.1. Characteristics of the considered examination type structure in the dataset, with additional details about the distribution of type nodes contained in the structure. Of the 596 nodes that form the examination type, 452 nodes are leaf nodes, which actually contain a value.

Note that, at the operational level, the complexity of the tree structure depends on the course of each specific examination.

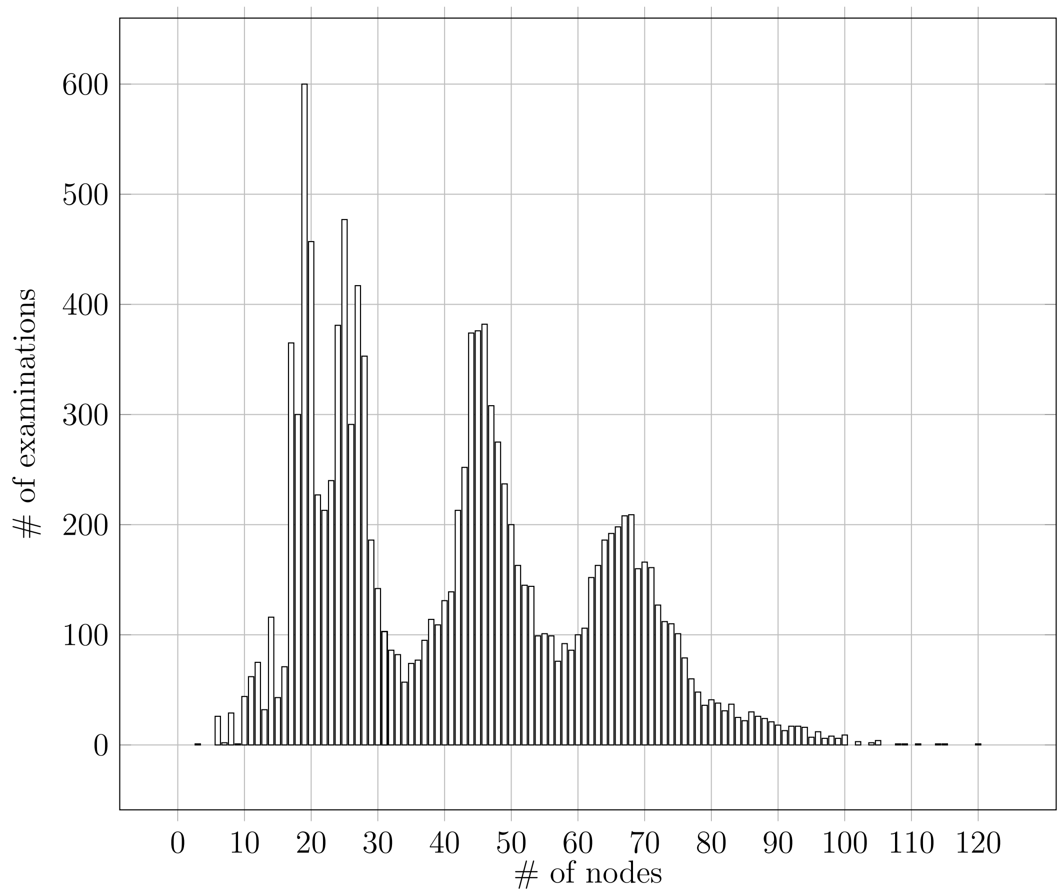


Figure 4.4. The histogram describes the distribution of examinations in the dataset as the number of nodes varies. Note that about 35% of the examinations in the dataset are in the neighbourhood of 23±6, with peaks in 19, 20, 25 and 27. This shows clearly how, usually, only a small part of the examination structure, that comprises a total number of 596 nodes, is actually filled out by health professionals. Only about 9% of examinations in the dataset have more than 70 nodes filled out.

Fig. 4.4 reports the distribution of examinations per number of nodes. Fig. 4.5 characterizes the distribution of examinations per depth of the tree structure. From these statistics, it is possible to note that the size of the tree-like structure (composed by `Facts`) is always much lower than the size of the corresponding graph structure (composed by `FactTypes`). This is due to the fact that, during a clinical session, not all the observations represented in the examination structure (≈ 600) are actually recorded.

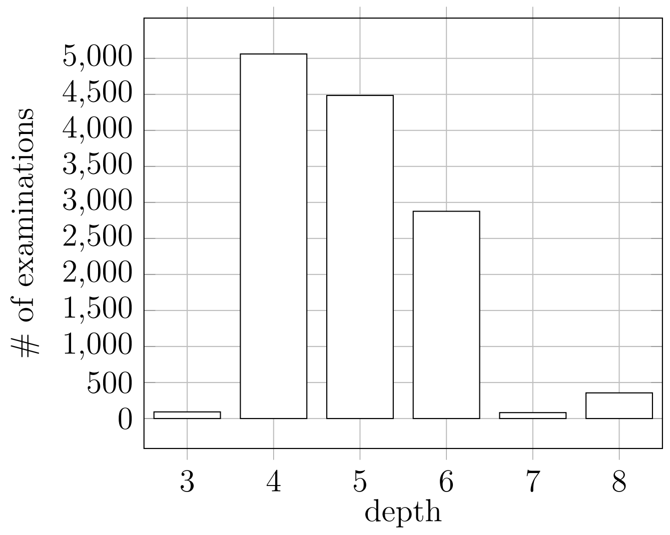


Figure 4.5. The histogram describes the distribution of examinations as the depth increases. Note that 96% of the examinations in the dataset have depth comprises between 4 and 6.

The *synthetic* dataset contains generated examinations with a full binary tree structure, with depth ranging from 2 to 8. For each depth, a fixed number of 100 examinations has been generated. Being a full binary tree, the number of nodes n for each tree of depth d is given by:

$$n = 2^{d+1} - 1,$$

ranging from 3 to 511 nodes.

The synthetic dataset does not correspond to a real situation in the present context-of-use of the *Empedocle* EHR system, but it can become a possible scenario in the evolution of the use of the system, and, for this reason, represents a relevant part of the motivation for this performance engineering investigation. In the more general perspective of a meta-modeling architecture, this corresponds to the case where different courses can be described on a structure with different degrees of completeness.

The experimentation on both datasets under consideration has been carried out with reference to a major scenario of interaction: a health professional accesses the patient's EHR content in order to review past medical examinations and read collected clinical information (Fig. 2.2 depicts the read-only use case in a typical outpatient scenario). To do that, each examination in the dataset has first been retrieved and, then, a read-only operation has been performed in order to simulate the real interaction of users with the EHR system through the interfaces exposed by the Presentation Layer.

As a metric of performance, we evaluated the total time required to complete the selected scenario, from data retrieving to data serialization, for all compared models. Specifically, we measure the time for completing the whole use case, comprising database retrieval operations, Java database APIs, or ORM persistence layer intermediation (only present in the relational case), without distinguishing time passed by the various phases of data retrieval and process. This is because experimental results not reported here indicate that these accessories operations, and in particular the ORM mapping layer (implemented by *Hibernate* in the current application stack), do not significantly impact the overall performance (e.g. Hibernate is optimized for the underlying database technology [83, 131]). Finally, note that retrieving an examination also implies retrieving the associated structure, containing the semantic of all `Facts` within the retrieved examination.

Experiments were conducted on a computer with the following characteristics: Debian 3.2.60 operating system, with 2 x Intel Xeon E5640 @ 2.66 GHz 64-Bit CPU, and 32 GB RAM.

4.2.4.2 Results

Table 4.2 and Fig. 4.6 report the results of the experimentation on the real dataset, showing the mean value (μ) and the coefficient of variation (CV) of the time spent to complete a read-only operation for a single examination in the three implementations under test. These statistical indexes were evaluated by repeating the read-only task for 100 times on all 12 953 examinations in the dataset. In comparison with the MySQL + Hibernate implementation, Neo4j reduces the retrieval time by approximately 1.5 times, and MongoDB reduces it by more than 33 times.

MySQL + Hibernate		**Neo4j**		**MongoDB**	
μ (ms)	CV	μ (ms)	CV	μ (ms)	CV
76.06	0.031	51.29	0.0024	2.27	0.064

Table 4.2. Comparison between MySQL+Hibernate, Neo4j, and MongoDB, evaluated using the *real* dataset comprising 12 953 examinations. Table reports mean value (μ) and coefficient of variation (CV) for the execution of a single examination.

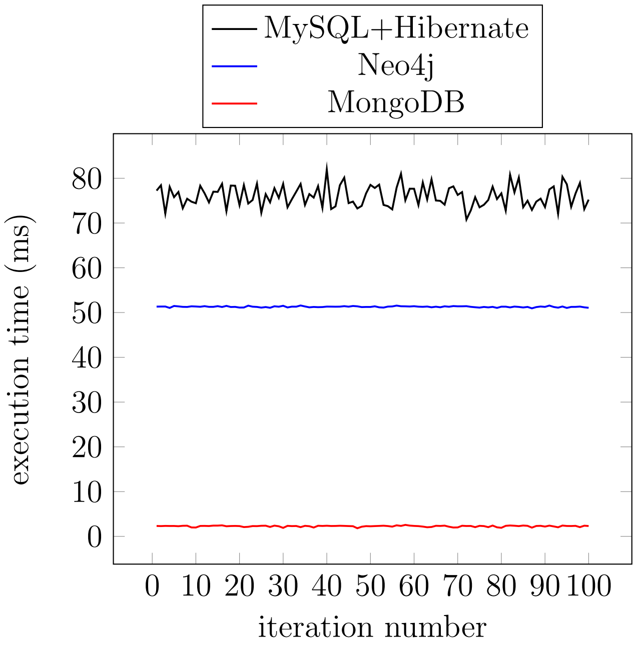


Figure 4.6. Performance results obtained by repeating the read-only task for 100 times on all 12 953 examinations in the *real* dataset.

Table 4.3 and Fig. 4.7 show the results of experimentation on the *synthetic* dataset. We report the mean value (μ) and the coefficient of variation (CV) of the time spent to complete a read-only operation for a single examination in the three implementations under test, evaluated by repeating the task for 100 times on all 100 examinations in the dataset. Results indicate that MongoDB attains by far a better performance and slower sensitivity to the examination depth. It should also be noted that the MySQL + Hibernate implementation performs better than Neo4j for examinations with depth lower than 7.

Depth	**MySQL + Hibernate**		**Neo4j**		**MongoDB**	
	μ (ms)	CV	μ (ms)	CV	μ (ms)	CV
2	6.93	0.12	18.76	0.12	1.07	0.05
3	9.54	0.11	19.41	0.09	1.2	0.06
4	12.6	0.1	21.57	0.09	1.38	0.07
5	18.87	0.09	26.04	0.08	1.64	0.07
6	28.17	0.09	33.94	0.05	2.2	0.07
7	48.18	0.08	44.03	0.05	3.05	0.08
8	121.29	0.04	72.93	0.05	4.88	0.07

Table 4.3. Comparison between MySQL+Hibernate, Neo4j, and MongoDB, evaluated using the *synthetic* dataset comprising 100 examinations with increasing depth. Table reports the mean value (μ) and the coefficient of variation (CV) for the execution of a single examination.

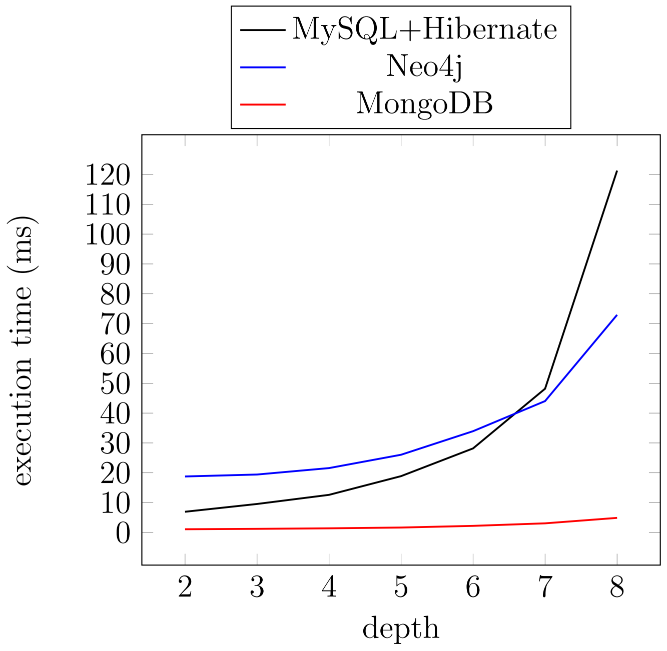


Figure 4.7. Performance results obtained by repeating the read-only task for 100 times on all the 100 examinations in the *synthetic* dataset.

4.2.4.3 Discussion

Performance results obtained during experimentations in *real* and *synthetic* datasets indicate a clear gain in performance through the use of MongoDB database, and more generally, a better scalability of NoSQL technologies as the depth of examinations grows, due to the increased number of JOINs and reference operations affecting the MySQL + Hibernate solution [114].

Moreover, both tested NoSQL technologies offer advantages in terms of flexibility in the data model, scalability and reliability.

Results also indicate a counter-intuitive conclusion: the graph-oriented data model of Neo4j allows a more natural and direct data conversion, which also permits a simpler implementation; however, the document-oriented data model of MongoDB produces by far better performance results. Specifically: Neo4j, which graph-oriented modeling is more natural in the considered software architectural context, shows a performance gain of 1.5 times compared to MySQL + Hibernate. On the other hand, MongoDB, which requires a bigger engineering investment to convert the original data model for achieving the right trade-off between redundancy, adaptability and reactivity, shows a performance gain of almost 33 times compared to MySQL + Hibernate.

The present investigation is completely open to explore the performance of NoSQL databases in other use cases, not only limited to read-only operations but also extended to write and update scenarios, whose impact in the context-of-use is less relevant but nonetheless interesting to have a full comparison between the various analyzed data models.

Conclusions

A digital universe of unstructured or semi-structured human-sourced information, structured process-mediated data, and well-structured machine-generated data is rapidly growing over the last decade, encouraging the adoption of innovative forms of data modeling and information processing to enable enhanced insight, decision making, and process automation applied to a variety of different contexts [19].

Healthcare comprises a notable domain of interest, due to the availability of a large amount of information that can be efficiently exploited to achieve some relevant and tangible benefits, including: improvement of operational efficiency and patient outcomes; early identification of comorbidities as well as worsening health states; evidence of effectiveness and safety of therapeutic strategies; reduction of health system costs; real enactment of predictive models for diagnosing, treating, and delivering care.

However, the health domain constitutes a context with intrinsic complexities, mainly derived by the nature of data to deal with, and whose characteristics can be summarized in terms of volume, variety, variability, velocity, and veracity [80]. This complex nature represents one of the main hurdles to be closely faced.

In this dissertation we focused on the crucial role played by software architectures in order to overcome *data modeling* and *information processing* challenges posed by the healthcare context. Specifically, we investigated the applicability of multi-level meta-modeling architectural styles in two different health scenarios, i.e. healthcare and Ambient Assisted Living (AAL) contexts.

Multi-level meta-modeling architectures [28, 134] represent a powerful alternative to consolidated object-oriented solutions that fit well the development of (static) systems characterized by a stable domain with a low rate of change but inevitably fail in volatile contexts.

Otherwise, a multi-level meta-modeling architecture supports the development of systems able to change structure and behavior dynamically, tailoring itself to the case of domains characterized by high volatility and complexity, where adaptability and changeability represent primary requirements [55].

Concerning how medical concepts and clinical data can be effectively and efficiently represented to fully exploit the large amount of available information (i.e. data modeling challenge), this dissertation addressed the design and implementation of an adaptable patient-centric Electronic Health Record (EHR) system, named *Empedocle* [88], facing a number of challenging requirements, including: adaptability to different specialities and organizational contexts; run-time configurability by domain experts; interoperability of heterogeneous data produced by various sources and accessed by various actors. As a result, the *Empedocle* EHR system is in use since more than 3 years in various units of the major hospital of Tuscany Region (Careggi Hospital, in Florence).

Concerning how all available data can be exploited for enabling advanced decision-making processes (i.e. information processing challenge), we expanded the research in two different directions.

First, in order to provide patient-specific advices based on existing clinical data and support automated compliance evaluation of the quality of clinical processes, we addressed the integration of recommendations extracted from medical guidelines with clinical data collected within the *Empedocle* EHR system via a two-step diagnostic process [89] consisting of: *i)* collecting clinical information using an unbiased general EHR structure; *ii)* automatically adapting the EHR content to the specific formulated diagnosis in accordance to related applicable guidelines. Preliminary experimentations on real guidelines have shown the feasibility and benefits of the proposed approach.

Secondly, due to the crucial role played by Ambient Assisted Living (AAL) technologies for providing assistance care at home and supporting elderly people in their daily activities, we investigated how an adaptable EHR system can be exploited to collect clinical data produced by human actors as well as sensed data generated by remote monitoring devices deployed in a pervasive intelligent environment. Than, we proposed a continuous-time model-based approach for Activity Recognition (AR) for monitoring and classifying high-level human activities starting from low-level sensed data collected with the EHR system [29]. The proposed approach was validated with reference to a public dataset widely used in applications of AAL [125], providing results that show comparable performance with state-of-the-art AR discrete-time approaches.

The development and applicability of meta-modeling architectures to different contexts of eHealth exposed the resulting adaptable systems to some weakness. Specifically, a problem of performance inefficiencies is emerged, due to the high degree of abstraction of the underlying meta-model, requiring extra processing and instantiation, at run-time, of an increased number of objects (and relationships) for describing the whole domain. We addressed how refactoring interventions of the data model based on promising NoSQL technologies can impact on performance issues with respect to consolidated relational data solutions.

Bibliography

[1] HL7 International. `http://www.hl7.org/`.

[2] MongoDB for giant idea. `https://www.mongodb.org/`.

[3] Neo4j the world's leading graph database. `http://neo4j.com/`.

[4] openEHR, an open domain-driven platform for developing flexible e-health systems. `http://www.openehr.org/`.

[5] Hande Alemdar and Cem Ersoy. Wireless sensor networks for healthcare: A survey. *Computer Networks*, 54(15):2688–2710, 2010.

[6] Scott Ambler. *Agile Database Techniques: Effective Strategies for the Agile Software Developer*. John Wiley & Sons, Inc., New York, NY, USA, 2003.

[7] Elvio G. Amparore, Peter Buchholz, and Susanna Donatelli. A Structured Solution Approach for Markov Regenerative Processes. In *Int. Conference on Quantitative Evaluation of Systems*, pages 9–24. Springer, 2014.

[8] Colin Atkinson and Thomas Kühne. Reducing accidental complexity in domain models. *Software & Systems Modeling*, 7(3):345–359, 2008.

[9] Paolo Atzeni, Giorgio Ausiello, Carlo Batini, and Marina Moscarini. Inclusion and equivalence between relational database schemata. *Theoretical Computer Science*, 19(3):267–285, 1982.

[10] Umut Avci and Andrea Passerini. Improving Activity Recognition by Segmental Pattern Mining. *IEEE Trans. on Knowledge and Data Eng.*, 26(4):889–902, 2014.

[11] Ezio Bartocci, Luca Bortolussi, and Guido Sanguinetti. Data-driven statistical learning of temporal logic properties. In *Proc. Int. Conf. Formal Modeling and Analysis of Timed Systems*, pages 23–37. Springer, 2014.

[12] David W. Bates. Ten Commandments for Effective Clinical Decision Support: Making the Practice of Evidence-based Medicine a Reality. *Journal of the American Medical Informatics Association*, 10(6):523–530, August 2003.

[13] Thomas Beale. Archetypes: Constraint-based domain models for future-proof information systems. In *OOPSLA 2002 workshop on behavioral semantics*, volume 105, 2002.

[14] Thomas Beale and Sam Heard. Archetype Definitions and principles. Technical report, The openEHR Foundation, 2007.

[15] Thomas Beale, Sam Heard, and David Lloyd. openEHR architecture overview. Technical report, The openEHR Foundation, 2008.

[16] Tim Benson. *Principles of health interoperability HL7 and SNOMED*. Springer Science & Business Media, 2012.

[17] Eta S. Berner. *Clinical Decision Support Systems*. Theory and Practice. Springer, 1999.

[18] Bernard Berthomieu and Michel Diaz. Modeling and Verification of Time Dependent Systems Using Time Petri Nets. *IEEE Trans. on Soft. Eng.*, 17(3):259–273, March 1991.

[19] Mark A. Beyer and Douglas Laney. The importance of âĂŸbig dataâĂŹ: a definition. *Stamford, CT: Gartner*, 2012.

[20] Shoaib M Bhatti, Zhaid H. Abro, and Farzana Rufabro. Performance evaluation of java based object relational mapping tool. *Mehran University Research Journal of Engineering and Technology*, 32(2):159–166, 2013.

[21] Andrea Bobbio, András Horváth, and Miklós Telek. Matching three moments with minimal acyclic phase type distributions. *Stochastic models*, 21(2-3):303–326, 2005.

[22] Andrea Bobbio and Miklós Telek. Markov regenerative SPN with non-overlapping activity cycles. *Int. Computer Performance and Dependability Symp.*, pages 124–133, 1995.

[23] Grady Booch. *Object Oriented Analysis & Design with Application*. Pearson Education India, 2006.

[24] José Bravo, Ramón Hervás, and Marcela Rodriguez. *Ambient Assisted Living and Home Care: 4th International Workshop, IWAAL 2012, Vitoria-Gasteiz, Spain, December 3-5, 2012, Proceedings*, volume 7657. Springer, 2012.

[25] Giacomo Bucci, Laura Carnevali, Lorenzo Ridi, and Enrico Vicario. Oris: a Tool for Modeling, Verification and Evaluation of Real-Time Systems. *Int. Journal of SW Tools for Technology Transfer*, 12(5):391 – 403, 2010.

[26] Giacomo Bucci, Valeriano Sandrucci, and Enrico Vicario. Ontologies and Bayesian Networks in Medical Diagnosis. In *System Sciences (HICSS), 2011 44th Hawaii International Conference on*, pages 1–8, 2011.

[27] Robert Buchholz, Claudia Krull, Thomas Strigl, and Graham Horton. Using Hidden non-Markovian Models to Reconstruct System Behavior in Partially-Observable Systems. In *Int. ICST Conf. on Simulation Tools and Techniques*, page 86. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010.

[28] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. *Pattern-oriented Software Architecture: A System of Patterns*. John Wiley & Sons, Inc., New York, NY, USA, 1996.

[29] Laura Carnevali, Christopher Nugent, Fulvio Patara, and Enrico Vicario. A Continuous-Time Model-Based Approach to Activity Recognition for Ambient Assisted Living. In *Quantitative Evaluation of Systems*, pages 38–53. Springer, 2015.

[30] Sudarshan S. Chawathe, Anand Rajaraman, Hector Garcia-Molina, and Jennifer Widom. Change Detection in Hierarchically Structured Information. *SIGMOD Rec.*, 25(2):493–504, June 1996.

[31] Chiehfeng Cliff Chen, Kung Chen, Chien-Yeh Hsu, and Yu-Chuan Jack Li. Developing guideline-based decision support systems using protégé and jess. *Computer methods and programs in biomedicine*, 102(3):288–294, 2011.

[32] Liming Chen, Jesse Hoey, Chris D Nugent, Diane J Cook, and Zhiwen Yu. Sensor-based activity recognition. *Systems, Man, and Cybernetics,*

Part C: Applications and Reviews, IEEE Transactions on, 42(6):790–808, 2012.

[33] Liming Chen, Chris D. Nugent, and Hui Wang. A knowledge-driven approach to activity recognition in smart homes. *Knowledge and Data Engineering, IEEE Transactions on*, 24(6):961–974, 2012.

[34] Hoon Choi, Vidyadhar G. Kulkarni, and Kishor S. Trivedi. Markov regenerative stochastic Petri nets. *Perf. Eval.*, 20(1-3):337–357, 1994.

[35] Gianfranco Ciardo, Reinhard German, and Christoph Lindemann. A characterization of the stochastic process underlying a stochastic Petri net. *IEEE Transactions on Software Engineering*, 20(7):506–515, 1994.

[36] Diane J. Cook, Juan C. Augusto, and Vikramaditya R. Jakkula. Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Comp.*, 5(4):277–298, 2009.

[37] Luigi P. Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. Performance evaluation of the vf graph matching algorithm. In *Image Analysis and Processing, 1999. Proceedings. International Conference on*, pages 1172–1177. IEEE, 1999.

[38] Christopher John Date. *An introduction to database systems*, volume 7. Addison-wesley Reading, MA, 1990.

[39] Marilyn J. Field, Kathleen N. Lohr, et al. *Clinical Practice Guidelines: Directions for a New Program*, volume 90. National Academies Press, 1990.

[40] Marilyn J. Field, Kathleen N. Lohr, et al. *Guidelines for Clinical Practice:: From Development to Use.* National Academies Press, 1992.

[41] Martin Fowler. *Analysis patterns: reusable object models.* Addison-Wesley, 1997.

[42] Ulrich Frank. Modeling products for versatile e-commerce platformsâĂŤessential requirements and generic design alternatives. In *Conceptual Modeling for New Information Systems Technologies*, pages 444–456. Springer, 2002.

[43] Daniel J. Friedman and R. Gibson Parrish. The population health record: concepts, definition, design, and implementation. *Journal of the American Medical Informatics Association*, 17(4):359–366, 2010.

[44] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software.* Pearson Education, 1994.

[45] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1979.

[46] Rick Goud, Arie Hasman, and Niels Peek. Development of a guideline-based decision support system with explanation facilities for outpatient therapy. *Computer methods and programs in biomedicine*, 91(2):145–153, 2008.

[47] Kristiina Häyrinen, Kaija Saranto, and Pirkko Nykänen. Definition, structure, content, use and impacts of electronic health records: a review of the research literature. *International Journal of Medical Informatics*, 77(5):291, 2008.

[48] Mark A. Hoffman and Marc S. Williams. Electronic medical records and personalized medicine. *Human genetics*, 130(1):33–39, 2011.

[49] Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns: Designing, building, and deploying messaging solutions.* Addison-Wesley Professional, 2004.

[50] A. Horváth and Miklós Telek. PhFit: A General Phase-Type Fitting Tool. In *Int. Conf. Computer Performance Evaluation, Modelling Techniques and Tools - TOOLS 2002*, pages 82–91, 2002.

[51] Andras Horváth, Marco Paolieri, Lorenzo Ridi, and Enrico Vicario. Transient analysis of non-Markovian models using stochastic state classes. *Performance Evaluation*, 69(7-8):315–335, 2012.

[52] Richard Hull. Relative information capacity of simple relational database schemata. *SIAM Journal on Computing*, 15(3):856–886, 1986.

[53] Ilias Iakovidis. Towards personal health record: current situation, obstacles and trends in implementation of electronic healthcare record in europe. *International journal of medical informatics*, 52(1):105–115, 1998.

[54] Christopher Ireland, David Bowers, Michael Newton, and Kevin Waugh. A classification of object-relational impedance mismatch. In *Advances in*

Databases, Knowledge, and Data Applications, 2009. DBKDA'09. First International Conference on, pages 36–43. IEEE, 2009.

[55] ISO. ISO/IEC9126:2001. Software engineering – Product quality. Technical report, International Organization for Standardization, 2001.

[56] ISO. ISO/TR20514:2005. Health informatics – Electronic health record – Definition, scope and context. Technical report, International Organization for Standardization, 2005.

[57] Peter Jackson. *Introduction to Expert Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 1998.

[58] Robert A. Jenders. Decision Rules and Expressions. *Clinical Decision Support: The Road to Broad Adoption*, page 417, 2014.

[59] Ralph Johnson and Jeff Oakes. The User-Defined Product Framework. *URL: http://st.cs.uiuc.edu/pub/papers/frameworks/udp*, 1998.

[60] Ralph Johnson and Bobby Woolf. The Type Object Pattern. pages 47–66. Addison-Wesley Professional, 1997.

[61] Emil Jovanov, Aleksandar Milenkovic, Chris Otto, and Piet C. De Groen. A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation. *Journal of NeuroEngineering and rehabilitation*, 2(1):6, 2005.

[62] Anna Jurek, Chris Nugent, Yaxin Bi, and Shengli Wu. Clustering-based ensemble learning for activity recognition in smart homes. *Sensors*, 14(7):12285–12304, 2014.

[63] Anuradha Kanade, Aarthi Gopal, and Shantanu Kanade. A study of normalization and embedding in mongodb. In *Advance Computing Conference (IACC), 2014 IEEE International*, pages 416–421. IEEE, 2014.

[64] Hajar Kashfi. The intersection of clinical decision support and electronic health record: a literature review. pages 347–353, 2011.

[65] Sidney Katz, Thomas D. Downs, Helen R. Cash, and Robert C Grotz. Progress in development of the index of adl. *The gerontologist*, 10(1 Part 1):20–30, 1970.

[66] Daniel L. Kent, Edward H. Shortliffe, Robert W. Carlson, Miriam B. Bischoff, and Charlotte D. Jacobs. Improvements in data collection through physician use of a computer-based chemotherapy treatment consultant. *Journal of Clinical Oncology*, 3(10):1409–1417, 1985.

[67] Eunju Kim, Sumi Helal, and Diane Cook. Human activity recognition and pattern discovery. *Pervasive Computing, IEEE*, 9(1):48–53, 2010.

[68] Igor Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine*, 23(1):89–109, 2001.

[69] Ilkka Korhonen, Juha Parkka, and Mark Van Gils. Health monitoring in the home of the future. *Engineering in Medicine and Biology Magazine, IEEE*, 22(3):66–73, 2003.

[70] Steven H. Landers. Why health care is going home. *New England Journal of Medicine*, 363(18):1690–1691, 2010.

[71] Ali Larab, Emmanuel Conchon, Rémi Bastide, and Nicolas Singer. A sustainable software architecture for home care monitoring applications. In *Digital Ecosystems Technologies (DEST), 2012 6th IEEE International Conference on*, pages 1–6. IEEE, 2012.

[72] Paulo J. Lisboa and Azzam F.G. Taktak. The use of artificial neural networks in decision support in cancer: a systematic review. *Neural networks*, 19(4):408–415, 2006.

[73] Peter Lucas. *Bayesian networks in medicine: a model-based approach to medical decision making.* na, 2001.

[74] John Lumpkin, Simon P. Cohn, Jeffrey S. Blair, et al. Uniform data standards for patient medical record information. *National Committee on Vital and Health Statistics*, 53, 2003.

[75] Ronny S. Mans, Helen Schonenberg, Min Song, Will M. P. van der Aalst, and Piet J. M. Bakker. Application of Process Mining in Healthcare - A Case Study in a Dutch Hospital. In *Biomedical Eng. Sys. and Technologies*, pages 425–438. Springer, 2009.

[76] Robert C. Martin. The dependency inversion principle. *C++ Report*, 8(6):61–66, 1996.

[77] Susan Mckeever, Juan Ye, Lorcan Coyle, Chris Bleakley, and Simon Dobson. Activity recognition using temporal evidence theory. 2010.

[78] Silvia Miksch, Yuval Shahar, and Peter Johnson. Asbru: a task-specific, intention-based, and time-oriented language for representing skeletal plans. In *Proceedings of the 7th Workshop on Knowledge Engineering: Methods & Languages (KEML-97)*, pages 9–19. Milton Keynes, UK, The Open University, Milton Keynes, UK, 1997.

[79] Carl D. Mitchell and Leah H. Jamieson. Modeling duration in a hidden Markov model with the exponential family. In *Int. Conf. Acoustics, Speech, and Signal Processing*, volume 2, pages 331–334. IEEE, 1993.

[80] Travis B. Murdoch and Allan S. Detsky. The inevitable application of big data to health care. *Jama*, 309(13):1351–1352, 2013.

[81] Marcel F. Neuts. *Matrix Geometric Solutions in Stochastic Models*. Johns Hopkins University Press, 1981.

[82] Vuyisile T. Nkomo, Julius M. Gardin, Thomas N. Skelton, John S. Gottdiener, Christopher G. Scott, and Maurice Enriquez-Sarano. Burden of valvular heart diseases: a population-based study. *The Lancet*, 368(9540):1005–1011, 2006.

[83] Elizabeth J. O'Neil. Object/relational mapping 2008: hibernate and the entity data model (edm). In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1351–1356. ACM, 2008.

[84] Elpiniki I. Papageorgiou, Jos De Roo, Csaba Huszka, and Dirk Colaert. Formalization of treatment guidelines using fuzzy cognitive maps and semantic web tools. *Journal of biomedical informatics*, 45(1):45–60, 2012.

[85] George Papamarkos, Alexandra Poulovassilis, and Peter T. Wood. Event-Condition-Action Rule Languages for the Semantic Web. In *Workshop on Semantic Web and Databases*, pages 309–327, 2003.

[86] Guido Parente, Christopher D Nugent, Xin Hong, Mark P Donnelly, Liming Chen, and Enrico Vicario. Formal modeling techniques for ambient assisted living. *Ageing International*, 36(2):192–216, 2011.

[87] Fulvio Patara, Chris D. Nugent, and Enrico Vicario. Recommendations for the Creation of Datasets in Support of Data Driven Activity Recognition Models. In *Inclusive Smart Cities and e-Health*, pages 79–91. Springer, 2015.

[88] Fulvio Patara and Enrico Vicario. An adaptable patient-centric Electronic Health Record system for personalized home care. In *Medical Information and Communication Technology (ISMICT), 2014 8th International Symposium on*, pages 1–5. IEEE, 2014.

[89] Fulvio Patara and Enrico Vicario. Dynamic Adaptation of EHR Structure for Automated Compliance Evaluation. *Studies in health technology and informatics*, 205:1238, 2014.

[90] Norman W. Paton and Oscar Díaz. Active database systems. *ACM Comput. Surv.*, 31(1):63–103, 1999.

[91] Donald J. Patterson, Lin Liao, Dieter Fox, and Henry Kautz. Inferring high-level behavior from low-level sensors. In *Ubiquitous Computing*, pages 73–89, 2003.

[92] Mor Peleg. Computer-interpretable clinical guidelines: A methodological review. *Journal of Biomedical Informatics*, pages 1–20, 2013.

[93] Mor Peleg, Aziz A. Boxwala, Omolola Ogunyemi, Qing Zeng, Samson Tu, Ronilda Lacson, Elmer Bernstam, Nachman Ash, Peter Mork, Lucila Ohno-Machado, et al. GLIF3: the evolution of a guideline representation format. In *Proceedings of the AMIA Symposium*, page 645. American Medical Informatics Association, 2000.

[94] Mor Peleg, Sagi Keren, and Yaron Denekamp. Mapping computerized clinical guidelines to electronic medical records: Knowledge-data ontological mapper (KDOM). *Journal of biomedical informatics*, 41(1):180–201, 2008.

[95] Mor Peleg, Yuval Shahar, and Silvana Quaglini. Making healthcare more accessible, better, faster, and cheaper: the MobiGuide Project. *European Journal of ePractice: Issue on Mobile eHealth*, 20:5–20, 2014.

[96] Stephanie E. Pollard, Pamela M. Neri, Allison R. Wilcox, Lynn A. Volk, Deborah H. Williams, Gordon D. Schiff, Harley Z. Ramelson, and David W. Bates. How physicians document outpatient visit notes in an

electronic health record. *International Journal of Medical Informatics*, 82(1):39–46, January 2013.

[97] Hossein Pourreza, Sergio Camorlinga, Carson K-S Leung, and Bruce D. Martin. An information and communication technology system to support rural healthcare delivery. In *Proceedings of the 1st ACM International Health Informatics Symposium*, pages 440–444. ACM, 2010.

[98] Daniel J. Power, Ramesh Sharda, and Frada Burstein. *Decision support systems*. Wiley Online Library, 2002.

[99] Silvana Quaglini, Mario Stefanelli, Giordano Lanzola, Vincenzo Caporusso, and Silvia Panzarasa. Flexible guideline-based patient careflow systems. *Artificial intelligence in medicine*, 22(1):65–80, 2001.

[100] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[101] Parisa Rashidi and Diane J. Cook. Keeping the resident in the loop: Adapting the smart home to the user. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 39(5):949–959, 2009.

[102] Parisa Rashidi, Diane J. Cook, Lawrence B. Holder, and Maureen Schmitter-Edgecombe. Discovering activities to recognize and track in a smart environment. *IEEE Transactions on Knowledge and Data Engineering*, 23(4):527–539, 2011.

[103] Sandra Regan and Sabrina T. Wong. Patient perspectives on primary health care in rural communities: effects of geography on access, continuity and efficiency. *Rural and Remote Health*, 9(1142), 2009.

[104] Philipp Reinecke, Tilman Krauß, and Katinka Wolter. Phase-Type Fitting Using HyperStar. In *Europ. Workshop on Computer Perf. Eng.*, pages 164–175, 2013.

[105] Dirk Riehle. A Role-Based Design Pattern Catalog of Atomic and Composite Patterns Structured by Pattern Purpose. Technical report, 1997.

[106] Dirk Riehle, Michel Tilman, and Ralph Johnson. Dynamic object model. *Pattern Languages of Program Design*, 5:3–24, 2000.

[107] Andrea Rogge-Solti and Mathias Weske. Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In *Service-Oriented Computing*, pages 389–403. Springer, 2013.

[108] Gillian D. Sanders, Robert F. Nease, Douglas K. Owens, et al. Design and pilot evaluation of a system to develop computer-based site-specific practice guidelines from decision models. *Medical Decision Making*, 20(2):145–159, 2000.

[109] Aaron Schram and Kenneth M. Anderson. MySQL to NoSQL: data modeling challenges in supporting scalability. In *Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity*, pages 191–202. ACM, 2012.

[110] Baron Schwartz, Peter Zaitsev, and Vadim Tkachenko. *High performance MySQL: Optimization, backups, and replication.* " O'Reilly Media, Inc.", 2012.

[111] Amnon Shabo. The implications of electronic health records for personalized medicine. *Personalized medicine*, 2(3):251–258, 2005.

[112] Richard N. Shiffman, Bryant T. Karras, Abha Agrawal, Roland Chen, Luis Marenco, and Sujai Nath. GEM: a proposal for a more comprehensive guideline document model using XML. *Journal of the American Medical Informatics Association*, 7(5):488–498, 2000.

[113] Edward H. Shortliffe. Computer programs to support clinical decision making. *Journal of the American Medical Association*, 258(1):61–66, 1987.

[114] Connie U. Smith and Lloyd G. Williams. Software performance antipatterns. In *Workshop on Software and Performance*, pages 127–136, 2000.

[115] Michael Stonebraker. Sql databases v. nosql databases. *Communications of the ACM*, 53(4):10–11, 2010.

[116] Ewa Straszecka. Combining uncertainty and imprecision in models of medical diagnosis. *Information Sciences*, 176(20):3026–3059, 2006.

[117] David R. Sutton and John Fox. The syntax and semantics of the PROforma guideline modeling language. *Journal of the American Medical Informatics Association*, 10(5):433–443, 2003.

[118] Paul C. Tang, Joan S. Ash, David W. Bates, J. Marc Overhage, and Daniel Z. Sands. Personal health records: definitions, benefits, and strategies for overcoming barriers to adoption. *Journal of the American Medical Informatics Association*, 13(2):121–126, 2006.

[119] Kishor S. Trivedi. *Probability and statistics with reliability, queuing, and computer science applications.* John Wiley and Sons, New York, 2001.

[120] Samson W. Tu, Mark A. Musen, et al. Modeling data and knowledge in the eon guideline architecture. *Studies in health technology and informatics*, (1):280–284, 2001.

[121] Alec Vahanian, Ottavio Alfieri, Felicita Andreotti, Manuel J. Antunes, Gonzalo Barón-Esquivias, Helmut Baumgartner, Michael Andrew Borger, Thierry P. Carrel, Michele De Bonis, Arturo Evangelista, et al. Guidelines on the management of valvular heart disease (version 2012). *European heart journal*, 33(19):2451–2496, 2012.

[122] Will M. P. van Der Aalst, A. Adriansyah, Ana K. A. de Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh C. Bose, Peter van den Brand, Ronald Brandtjen, Joos Buijs, et al. Process mining manifesto. In *Business process management workshops*, pages 169–194. Springer, 2012.

[123] Will M. P. van der Aalst, Hajo A. Reijers, Ton A. J. M. M. Weijters, Boudewijn F. van Dongen, Ana K. A. De Medeiros, Min Song, and Eric H. M. W. Verbeek. Business process mining: An industrial application. *Information Systems*, 32(5):713–732, 2007.

[124] Boudewijn F. van Dongen, Ana K. A. de Medeiros, Eric H. M. W. Verbeek, A. J. M. M. Weijters, and Will M. P. van Der Aalst. The ProM framework: A new era in process mining tool support. In *Applications and Theory of Petri Nets 2005*, pages 444–454. Springer, 2005.

[125] Tim van Kasteren, Athanasios Noulas, Gwenn Englebienne, and Ben Kröse. Accurate Activity Recognition in a Home Setting. In *Proc. Int. Conf. on Ubiquitous Computing*, UbiComp '08, pages 1–9, New York, NY, USA, 2008. ACM.

[126] Enrico Vicario, Luigi Sassoli, and Laura Carnevali. Using stochastic state classes in quantitative evaluation of dense-time reactive systems. *IEEE Transactions on Software Engineering*, 35(5):703–719, 2009.

[127] Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, and Dawn Wilkins. A comparison of a graph database and a relational database: a data provenance perspective. In *Proceedings of the 48th annual Southeast regional conference*, page 42. ACM, 2010.

[128] Ward Whitt. Approximating a Point Process by a Renewal Process, I: Two Basic Methods. *Operations Research*, 30(1):125–147, Jan.-Feb. 1982.

[129] Rebecca Wirfs-Brock, Brian Wilkerson, and Lauren Wiener. Designing object-oriented software. 1990.

[130] Steven H. Woolf, Richard Grol, Allen Hutchinson, Martin Eccles, and Jeremy Grimshaw. Clinical guidelines: potential benefits, limitations, and harms of clinical guidelines. *British Medical Journal*, 318(7182):527, 1999.

[131] Qinglin Wu, Yanzhong Hu, and Yan Wang. Research on data persistence layer based on hibernate framework. In *Intelligent Systems and Applications (ISA), 2010 2nd International Workshop on*, pages 1–4. IEEE, 2010.

[132] Juan Ye, Simon Dobson, and Susan McKeever. Situation identification techniques in pervasive computing: A review. *Pervasive and mobile computing*, 8(1):36–66, 2012.

[133] Joseph W. Yoder, Federico Balaguer, and Ralph Johnson. Architecture and design of adaptive object-models. *ACM Sigplan Notices*, 36(12):50–60, 2001.

[134] Joseph W. Yoder and Ralph Johnson. The adaptive object-model architectural style. In *Software Architecture*, pages 3–27. Springer, 2002.

[135] Armin Zimmermann. Dependability Evaluation of Complex Systems With TimeNET. In *Proc. Int. Workshop on Dynamic Aspects in Dependability Models for Fault-Tolerant Systems*, pages 33–34, 2010.